NISTIR 5636

Persistent Object Base System Testing and Evaluation

Elizabeth Fong

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

NIST

NISTIR 5636

Persistent Object Base System Testing and Evaluation

Elizabeth Fong

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

April 1995

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

PREFACE

The Computer Systems Laboratory (CSL) of the National Institute of Standards and Technology (NIST) is assisting the Advanced Research Projects Agency (ARPA) in a technology evaluation of persistent object bases (POBs). POBs, generally referred to as Object Database Management Systems (ODBMSs), are database systems which provide storage and retrieval of data that are not necessarily tabular (with rows and columns).

This project was conducted as part of the ARPA Persistent Object Base Project, ARPA Order No. 8217. Its publication as a NISTIR does not imply ARPA endorsement of the conclusions or recommendations presented.

Certain commercial software products and companies are identified in this report for purposes of specific illustration. Such identification does not imply recommendation or endorsement by NIST, nor does it imply that the products identified are necessarily the best ones available for the purpose.

ABSTRACT

This report summarizes the role of the Computer Systems Laboratory (CSL) of the National Institute of Standards and Technology (NIST) in support of the Advanced Research Projects Agency (ARPA) in the testing and evaluation of persistent object base (POB) systems. The actual evaluation consists of designing a testing suite for exercising the POB system for the features supported. The goal of actual testing is to reveal how well each feature is being supported. The testing methodology, the abstract test suite, and the actual test results on the ARPA funded prototype POB system called Open OODB developed by Texas Instruments are described.

Keywords: Evaluation; features; object; object Database Management; persistent object base system; test.

ACKNOWLEDGEMENTS

The ARPA POB project was started in 1990 by Erik Mettala who initially jump-started the POB evaluation and testing effort.

The major portion of the CSL's contribution to the POB project is managed under the direction of Dr. Gio Weiderhold. I would like to acknowledge Dr. Weiderhold for providing me with valuable technical direction into the evaluation of the POB technology and for his support in accelerating object-related standards.

K.C. Morris of the Factory Automation Systems Division is the co-principle investigator for this POB technology evaluation project. I would like to acknowledge her efforts in POB technology evaluation for engineering and manufacturing applications using the international Standard for Exchange of Product Model Data (STEP) and the STEP data access interface (SDAI).

The actual testing is conducted on an ARPA-funded POB system developed by Texas Instruments (TI), called the Open OODB. I would like to thank Dr. Deyuan Yang of NIST who performed the coding and execution of the test programs.

TABLE OF CONTENTS

1.0 INTRODUCTION

User acceptance of Persistent Object Base (POB)[1] technology is still weak. In 1993, the total object database market was approximately $32 million while ORACLE, a relational database vendor, alone makes $1.2 Billion. Due to a multiplicity of emerging object-related "standards" the general users are at the crossroads between a relational and an object database market.

This report summarizes the role of the Computer Systems Laboratory (CSL) of the National Institute of Standards and Technology (NIST) in support of the Advanced Research Projects Agency (ARPA) in the testing and evaluation of persistent object base (POB) systems. The goals of this project are to evaluate various aspects of POB technology, to use these evaluation results to assess applicability of object technology to design problems and then to identify the potential usefulness of object technology in a distributed environment.

1.1 Background

Object Database Management Systems (ODBMS) have emerged as a significant force during the last several years. There are two driving forces behind the emergence of ODBMS: the need to supply storage for persistent objects and the need to expand the scope of existing databases to support manufacturing and engineering applications.

A number of commercial ODBMS products and research prototypes have appeared. The Ovum Ltd. "Object Technology Sourcebook" [JEFF91] lists 9 vendors that offer ODBMSs that provide most database functionality. Additionally, [CATT91] covered a dozen more ODBMS products and prototypes. There are many research prototypes being funded by ARPA, e.g., Texas Instruments' Open OODB. The Object Database Management Group (ODMG) has published a Object Database Standard: ODMG-93 [CATT94]. There are several object database vendors promising products in the next year or two. The use of these products will expand, especially in the manufacturing and engineering sectors.

As with all new technology, no specific set of criteria currently can be applied to evaluate and thus identify a system as being "object." The purposes of an evaluation of POB technology are to:

- enable the adoption of new technology to industry and

---

[1] Although some believe that there are differences between a POB and an Object Database Management System (ODBMS), for this report, these two terms have been used interchangeably.

standards communities by providing evaluation and recommendations regarding the technology,

- provide a set of criteria for evaluating persistent object base systems,
- insure that developing persistent object systems are capable of supporting the needs of users in manufacturing and engineering,
- identify areas where standardization could accelerate the development of this technology.

1.2 The Scope of the POB Project

The overall scope of the NIST/ARPA POB project is very broad: to facilitate the transition of POB technology into industrial implementation. To accomplish this, two aspects of this process are targeted:

- transfer realistic and testable object database requirements to the POB community, and enable the transfer of new POB technology to the user community;
- generate recommendations for the definition of standards related to POB technology.

The specific task in support of the overall goals of the POB project is to develop a test and evaluation methodology. The actual evaluation involves not only a review of literature, POB products and documentation, but also an in-depth investigation on whether a POB system supports particular features. The evaluation consists of designing a test suite for exercising the POB system for the features supported. The goal of actual testing is to reveal "how well" each feature is being supported.

This report will describe the testing and evaluation of POB technology using one of the ARPA funded prototype POB systems, called Open OODB, developed by Texas Instruments [TIAP93, TIQU93].

1.3 Outline of this Report

An overview of the methodologies for testing is presented in Section 2. The actual testing strategy used for the evaluation of POBs is feature testing which is designed to test functionality and not its implementation. The feature areas and the testing process are discussed in Section 3. Section 4 contains the abstract test cases. Section 5 contains conclusions focused on TI's Open OODB and the results from the tests. The Appendix contains the TI Open OODB test environment description, a sample application diagram, and the actual executable test programs for Open OODB.

2.0 METHODOLOGIES FOR TESTING

The "test" addressed in this paper does not referred to find errors in a computer program, but rather used in the sense of finding the intended behavior of the software. In this context, many methodologies often associated with object-oriented software testing are not appropriate. The type of testing methodologies reviewed in this section include functional tests, performance tests, conformance tests, etc. An extended, formal treatment of test and verification methodology is beyond the scope of this paper; however, some testing methodologies are described here.

2.1 Functional Testing

Functional testing is also called quality testing. It is designed to determine if a specified feature, or capability is present in an implementation. Hence, the purpose of functional testing is for existence, correctness, and power of each feature. This means such a test is intended to show how well the functionality is being met.

Functional testing is useful for the ARPA/NIST POB testing and evaluation for several reasons:

- Functional testing will reveal the exact definition and behavior of a specific feature, irrespective of how it is implemented.

- Functional tests will guide users in articulating application requirements.

- The test results will permit users to verify that a given feature exhibits the desired functionality, thereby providing a level of quality assurance.

- The functional testing process will support the development of the rapidly evolving standard specifications. The initial specification of object features in the OODBTG Reference Model [OODB91] must be validated as meeting the needs of the user community. The results and recommendations generated by functional testing will be fed back to standards organizations for review and action. A standards committee may then amend the specifications, affected portions may be re-tested, and the specifications can be further improved.

It is important to realize that not all features are testable. Accordingly, it is important in the formulation of a list of features to determine an appropriate level of detail at which the features should be specified. A feature should be atomic enough that it makes some sense to assign a scalar (one-dimensional) rating, but not so detailed as to exhaust test resources (time and

personnel) in an attempt to evaluate a very large number of "small" factors.

The challenge in functional testing is that some functions may not be sharply defined, e.g., does everyone agree what "multiple inheritance" is? If the definition of function is not clear and subject to different interpretation, the test needs to supplement it with clarification.

Another consideration of functional testing is that the scoring technique for each test is largely manual. Testing if a capability exists should result in a "YES" or "NO" answer. However, to determine to what level a feature has been met requires human evaluation and relative ratings. Subjective numeric quantity can be used.

2.2 Performance Testing

A performance benchmark consists of a set of test programs designed to measure the performance of an operational system. Measuring performance of object database systems in a generic manner is very difficult, since every application has somewhat different requirements. For example, measuring individual persistent data object access at this level may provide some timing comparisons, but may not be accurate when measured in conjunction with an object query language processor.

The design of a benchmark for POB systems requires more research. In particular, the interpretation of benchmark results must be carefully defined. Assumptions on the data model, database architecture, programming languages used, etc. are all factors to be determined. Representative operations for engineering applications must be emphasized. A well known, simple, engineering database benchmark has been designed by R. Cattell [CATT91].

The design and execution of benchmark testing is a complex task. Problems with performance benchmark testing are quoted from [STON88] as follows:

> "Any person who designs a benchmark is in a 'no win' situation, i.e., he can only be criticized. External observers will find fault with the benchmark as artificial or incomplete in one way or another. Vendors who do poorly on the benchmark will criticize it unmercifully. On the other hand, vendors who do well will likely say the benchmark is poor but that one should use it anyway."

2.3 Conformance Testing

Conformance testing is designed to reveal whether the implemented system is in conformance with the standard specification. Of course, the first requirement is that the standard specification, be it formal or informal, must exist.

Conformance testing usually tests for both standardized syntax and semantics. Some standards also include test assertions for the construction of test suites. An example of abstract test suites for use in conformance testing is the ISO 10303 "Product Data Representation and Exchange [PDES94].

Conformance testing should have a predefined expected result, so that scoring in the form of pass/fail can be made automatically.

Since no standard currently exists for POB technology, no conformance testing is possible at this time.

2.4 Scalability or Stress Test

Another testing methodology is testing for scalability. Scalability testing is designed to find out how large can various objects (entities, attributes, relationships, etc.) be made and still be manipulated? How many simultaneous users' accesses can be supported? The purpose of scalability testing is to develop tests for maximum capacity and test for failure when maximum capacity is reached. However, the execution of some of the tests can be very time consuming and resource intensive. Most systems provide parameters for the scalability of some size definitions at system installation time, and most commercial systems allow very large, dynamically adjusting spaces.

2.5 Other Testing Methodologies

There are other testing methodologies designed to determine specific criteria of a system. For example, select representative usage scenarios of the intended applications may be defined. The objective of this kind of test is to determine if the system is suitable in performing the intended tasks for a particular domain. The evaluation criteria to be used in this area of testing need to be carefully determined. The use of application development tools also should be considered.

3.0 FUNCTIONAL TESTING OF FEATURES

The testing strategy for the NIST/ARPA POB project will be to evaluate and test the ability of POBs to support features which are widely required [FONG91]. The considerations for features to be tested are discussed in Section 3.2. The strategy is designed to test functionality (or capability), not its implementation. This means that, for instance, we may wish to know if location-transparent access to objects is possible, not the details of whether global data dictionaries are stored at particular sites or at all sites.

3.1 Determining How to Evaluate POB Features

The main focus of the design of POB tests is to determine which features are meaningful and testable.

As the project name "persistent object base" implies, some features to be tested are those typically associated with object data management. There is currently no consensus on a specific set of criteria that can be applied in order to evaluate and thus identify a POB system as being "object." There are many articles found today in the open literature which identify some of the existing and ongoing work in the definition of features needed to support "object" systems [DABR90, OODB91, STON90]. In the Object-Oriented Database Task Group report [OODB91], the reference model also identified a set of features for a database system to qualify as "object."

Certain features are simple and it is not necessary to design tests for them. An example of a simple feature might be whether the target system supports an ad-hoc query capability. Other features are not testable without exercising quantitative judgment. An example might be the "encapsulation" feature. Some features can be tested directly; however, the methods for testing can be diverse. For example, to design a test for concurrency control may require setting-up multiple user workstations and synchronizing the test execution. Some features can be executed interactively and the results are presented on the screen. In this case, the tester needs to eyeball the results and verify their correctness. Some features are best tested in compiled mode while some features need to be tested in interpreted mode.

A good policy for the design of feature tests is to develop independent tests which only exercise a single feature. Each test should be self-evaluating, i.e., each test is written with knowledge of the data in the database or the correct response for a specific statement. Each test checks for correct execution and prints out "pass" or "fail" value for that test.

There are feature areas that will be excluded from testing either because they are less critical for application development or because they are less understood or too difficult to test. These could include such things as methods for optimization. There are also many meta-features such openness, seamlessness, robustness, interoperability, etc., which will require more research to determine how they should be tested.

The abstract test suites defined in Section 4 of this report is a subset of what might be needed to totally evaluation a POB system.

3.2 Criteria For Features to be Tested

It is important to identify the criteria for selecting those features to be tested. Considerations for features to be tested are as follows:

- The features that are normally claimed to be "object" features. Examples of object features are persistence, object identities, inheritance, complex objects, etc.

- The features that are supported by more traditional database management systems. Examples of such features are schema definition, populating a database, data manipulation in the form of data insertion, data deletion, and data retrievals with conditions, etc.

- The advanced features that are required in support of engineering and manufacturing application development. These include support for version control, concurrency control, advanced and long transaction processing, complex data structures, schema evolution, distributed client-server architecture and communications, etc.

- There are other features which involve system integration aspects. For example, support for replication of data, or graphical user interfaces, etc.

3.3 The Testing Process

Testing will be done using test scripts (scenarios) run using fragments of data. Fragments of data consist of an object-oriented schema description (including operations) and instance data. The actions associated in one script may test many different features of the POB. Several scripts may be run using one data fragment.

The testing process involves the following steps:

- Identify feature areas that are generally considered to be useful for the POB.

- Identify a specific capability within a feature area which can be reasonably tested.

- Establish verdict criteria and conditions for the design of direct testing of this feature.

- Develop an abstract test case for this feature. This is the detailed specification of a scenario and the sequence of operations that needs to be performed and the specification of how the results are to be interpreted.

- Identify the implementation specific characteristics of the system under test.

- Translate the abstract test case scenarios into executable test code for the target POB system under test.

- Implement and run the executable test code with all the environment and conditions against the POB system to be tested.

- Compare and review the results with the predicted results. The tests are designed to isolate individual features as much as possible. It is quite possible that for each test case scenario, many features will be tested with coordinated sequences. The results may be accumulative with the previous results affecting the later results.

- Generate the validation report.

Figure 3.1 describes the testing process.

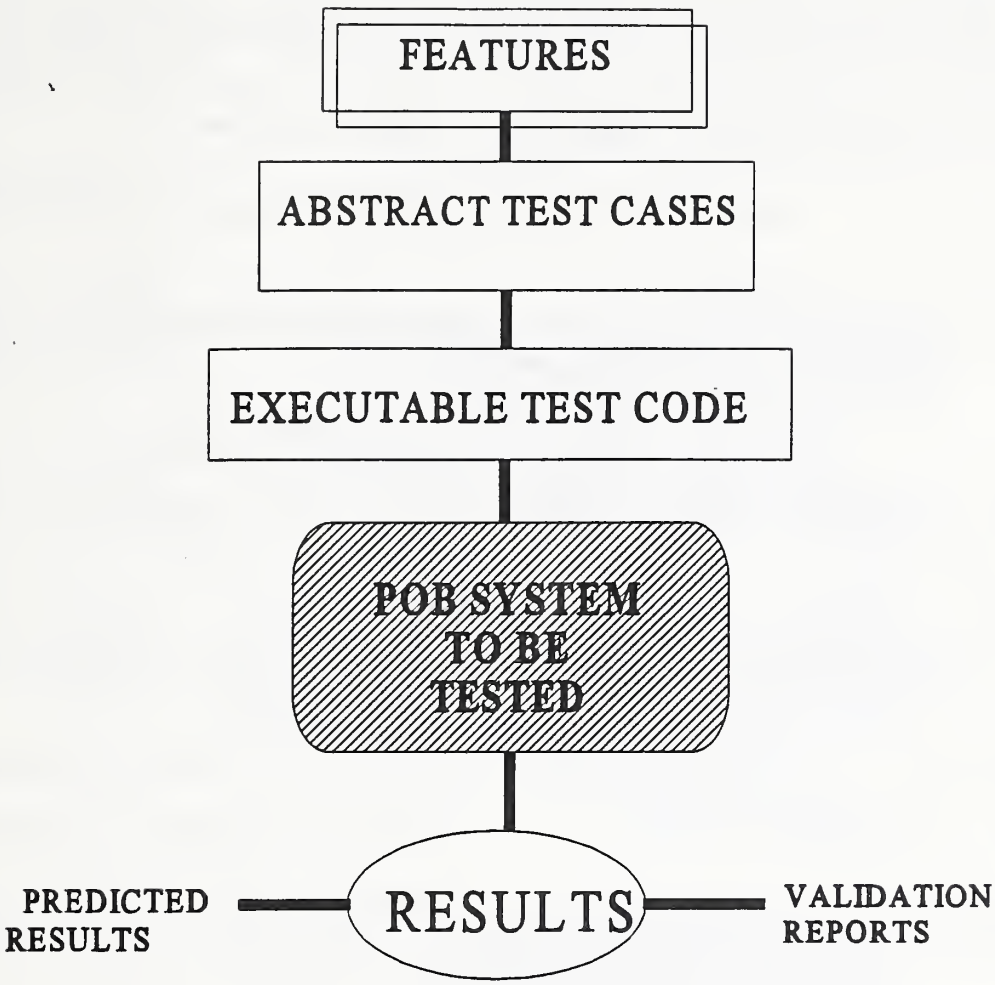


Figure 3.1 - Testing Process

4.0 ABSTRACT TEST SUITE DESCRIPTION

The abstract test suite for testing and evaluating a POB system consists of a set of abstract test cases. Each abstract test case is a specification, encapsulating at least one test purpose, that provides the basis from which executable test cases are derived.

4.1 Feature Groups of Abstract Test Suite

The abstract test cases are grouped according to the following functions:

- Simple Database Features:
    - Schemas or classes and sub-classes creation
    - Database population features
    - Data retrieval features:
        - Retrieval using object identifiers
        - Retrieval using object query language
    - Database update features
    - Meta data inspection and retrieval features

- Object Features:
    - Single inheritance features
    - Multiple inheritance features
    - Object identification retrieval (same as data retrieval using object identifiers).

- Advanced Database Features:
    - Version control
    - Concurrency control

- Advanced Features to Support Engineering Applications:
    - Configuration Management
    - Dynamic Schema Evolution
    - Distributed Communications
    - Recovery and Backup
    - Composite objects
    - Long Transaction Management
    - Authentication and Security

For this report, the advanced features to support engineering applications have not been developed. The above set of features identified may not be the complete test suites to totally evaluation a POB system under test.

4.2 Abstract Test Cases

The abstract test cases are described by the following set of headings: The test number, a prose description about the purpose of the test, the test requirements if any, and the expected results.

The test requirements are the pre-conditioning for this test case. The expected results, sometimes referred to as verdict criteria, are assertions on the observable output of an implementation under test.

The test application schema and data is also specified in a generic sense. The actual executable test code for exercising the target POB under test will use a sample application called the "Movie" database which is described in Appendix B.

Schemas or Classes and Subclasses Creation

Test Number: BDBF-001
Description: Create a schema with classes which have attributes.
Requirements: A schema could consist of one or more attributes.
Expected Result: A class with attributes is created.

Test Number: BDBF-002
Description: Create a schema with class and a sub-class.
Requirements: One class is a sub-class of the other class.
Expected Results: A class and a sub-class are created.

Test Number: BDBF-003
Description: Create multiple classes and sub-classes.
Requirements: Multiple classes with attributes and sub-classes with attributes are created. Attributes from super-classes are inherited.
Expected Results: The total schema is created.

Database Population Features

Test Number: BDBF-004
Description: Populate data into database with a batch file.
Requirements: An external batch file of data to be read and the schema of the database is defined.
Expected Results: Data loaded into database.

Data Retrieval Features

Test Number: BDBF-005
Description: Retrieve objects using record keys.
Requirements: One or more objects stored as persistent data.
Expected Results: Record with the designated key is retrieved.

Data Retrieval using Object Identifiers

Test Number: BDBF-006
Description: Retrieve objects using user-generated sequence number.

Requirements: One or more objects stored as persistent data.
Expected Results: The designed number of objects starting with specific sequence number are retrieved.

Test Number: BDBF-007
Description: Retrieve system-assigned Object Identifier (OID).
Requirements: Object Identifiers are assigned to objects stored as persistent data.
Expected Results: The OID is fetched and reported.

Form Collection of Data for Object Query Language

Test Number: BDBF-009
Description: Create a set or list of a class.
Requirements: Create a collection of objects: set or list. This is necessary for executing Object Query Language.
Expected Results: A set (list) of class is created.

Retrieval with Object Query Language

Test Number: BDBF-010.1
Description: Select objects using OQL where COMPARE_EQUAL.
Requirements: A collection of objects are formed such that Select-From-Where statement can be applied.
Expected Results: Matched results are reported.

Test Number: BDBF-010.2 and BDBF-010.3
Description: Select objects using OQL where LESS/GREATER.
Requirements: A collection of objects are formed such that Select-From-Where statement can be applied.
Expected Results: Less/Greater results are reported.

Test Number: BDBF-010.4 and BDBF-010.5
Description: Select objects using OQL qualified with AND/OR.
Requirements: A collection of objects are formed such that Select-From-Where statement with conjunction and disjunction can be applied.
Expected Results: Matched results are reported.

Database Update Features

Test Number: BDBF-011
Description: Read, modify and write objects.
Requirements: Known objects to be read, its values can be modified and stored back in persistent store.
Expected Results: Objects value changed to the modified values.

Meta Data Inspection and Retrieval Features

Test Number: BDBF-012
Description: Retrieve a specific class definition from dictionary.

Requirements: The "Type" information based upon the defined schema exists in the dictionary.
Expected Results: The type information is reported.

Test Number: BDBF-013
Description: Retrieve a class definition and all its sub-class definitions from dictionary.
Requirements: All of class definitions and its sub-class type and attribute information exit in the dictionary.
Expected Results: The class and sub-classes' definition including name and type information are reported.

Single Inheritance Features

Test Number: BDBF-015
Description: Create a class and a sub-class and allow all the attributes to be inherited to the sub-class.
Requirements: The attributes from the super-class are defined.
Expected Results: All the attributes are included in the sub-class.

Multiple Inheritance Features

Test Number: BDBF-016
Description: Create a sub-class which is multiply inherited from two super-classes. These two super-classes are in turn derived from the same super-super-class.
Requirements: The attributes from both super-classes are defined.
Expected Results: The dictionary should contain all the class relationships and attribute type information.

Version Control

Test Number: BDBF-019
Description: Create objects with version.
Requirements: New objects with version exists.
Expected Results: Display the newly versioned object.

Test Number: BDBF-020
Description: Fetch newly versioned objects and show version.
Requirements: Create a new version (do Test Number BDBF-19) and retrieve the newly versioned object and display the new version number.
Expected Results: Objects with new version indication.

Test Number: BDBF-21
Description: Fetch the old versioned objects which have been newly versioned.
Requirements: Create a new version (do Test Number BDBF-19) and retrieve the old versioned object.
Expected Results: Objects with old version indication.

Concurrency Controls

Test Number: BDBF-022
Description: Test for concurrency control when both clients are trying to read the same object.
Requirements: Start running a "fetch object" program and at another window initiate a "fetch object" program.
Expected Results: No standard way for concurrency control. Expect both programs are reading data or one of the read is lockout.

Test Number: BDBF-023
Description: Test for concurrency control when one client is retrieving, while the other client is trying to update the same object.
Requirements: Start running an "update object" program while also at another window initiate a "fetch object" program.
Expected Results: No standard way for concurrency control. Expect one program is reading and the other program is going to update. The read program should not be locked.

Test Number: BDBF-024
Description: Test for concurrency control when one client is updating, while the other client is trying to retrieve the same object.
Requirements: Start running a "fetch object" program while also at another window initiate an "update object" program.
Expected Result: No standard way for concurrency control. Expect the "retrieve object" should not get old data.

Test Number: BDBF-025
Description: Test for concurrency control when two clients are both trying to update the same object.
Requirements: Two clients updating one object in database concurrently.
Expected Results: One is updating and the other should wait until lock is released.

5.0 CONCLUSIONS

In this report, we have defined a number of feature test cases. These test cases are described abstractly in English. These test cases are then translated into executable code that can be run against the system under test; in this case, TI's Open OODB code. Each of the test programs is individually executed and results generated. The test programs and the results of Open OODB appeared in Appendix C of this report.

The Open OODB is designed as an open, modular, extensible architecture. It is essentially a "toolkit" in which modules in the toolkit can be combined in different ways so that custom systems can be built from reusable parts. The alpha release of the Open OODB system has minimal support for functionality including persistent object store, object query, change management and transaction store, but not extended transactions and other complex database functions. These tests were run on the alpha release of Open OODB. Later enhanced version of Open OODB may get different results.

The results obtained in this report reflect the observed behavior of the Open OODB with respect to a feature definition. The results are not indicated as "yes" or "no" because the semantics of features can be interpreted widely. For example, in testing for support of version control, one of the test case calls for the retention of old version, however, the system under test may not automatically support this feature. Testing the implementation of how concurrency control is supported can be tricky because there is no standard or "correct" way in specifying how the concurrency control feature should be implemented. The results analysis for feature testing cannot be automated into a "Pass" or "Fail" score, but need to be carefully analyzed.

For evaluation purposes, these abstract test cases may be re-used to test several different POB systems. Each feature area's test cases can be coded and executed on different POB systems and the results analyzed and compared. The feature analysis resulting from actual system testing could provide a greater level of precision in understanding the different semantics of a feature.

6.0 REFERENCES

[CATT91] Cattell, R.G.G. Object Data Management: Object-oriented and extended relational database systems, Addison-Wesley, 1991.

[CATT94] Cattell R.G.G. (Editor) The Object Database Standards: ODMG - 93, Morgan Kaufmann Publishers, 1994.

[DABR90] Dabrowski, C., Fong, E., and Yang, D., Object Database Management Systems: Concepts and Features, NIST Special Publication 500-179, April 1990

[FONG91] Fong, Elizabeth, "The ARPA/NIST Persistent Object Base Testbed," Position Paper for Texas Instruments Open OODB Workshop I, March 1991.

[JEFF91] Jeffcoate, J. and Templeton, A. Object Technology Sourcebook, Ovum Ltd, 1991.

[OODB91] Object-Oriented Database Task Group, "X3/SPARC/DBSSG/OODBTG Final Report, Available form E. Fong at efong@nist.gov September 1991.

[PDES94] ISO TC184/SC4/WG6 N82, "Guidelines for the Development of Abstract Test Suites," Working Draft of Product Data Representation and Exchange of ISO 10303 Part 1200 Series, 13 September 1993.

[STON88] Stonebraker, M. (ed.) Readings in Database Systems, Chapter 4, Performance and Database Machines, Morgan Kaufmann Publishers, Inc. 1988.

[STON90] Stonebraker, M. et. al., Committee for Advanced DBMS Functions, "Third-Generation Database System Manifesto," in SIGMOD Record, Volume 19, No. 3, September 1990.

[TIAP93] Open OODB C++ API User Manual Release 0.2 (Alpha), Texas Instruments Incorporated, 1993.

[TIQU93] Open OODB Query Language User Manual Release 0.2 (Alpha), Texas Instruments Incorporated, 1993.

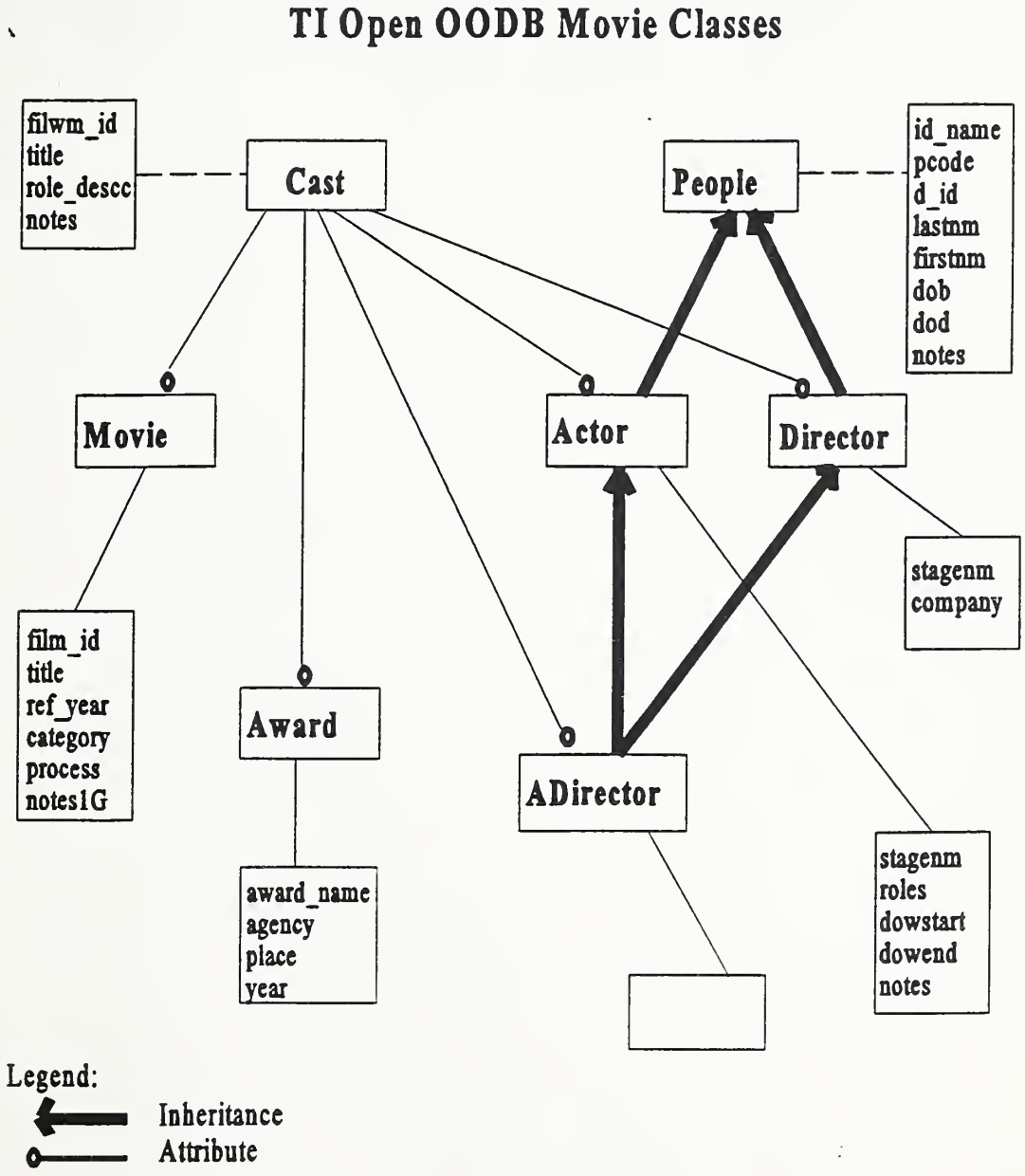
APPENDIX A - TI Open OODB Environment for Test

The Texas Instruments Open OODB (Release 0.2 Alpha) is the POB system under test. It is installed at the NIST Computer Systems Laboratory. The environment consists of the following:

Hardware: Sun/Sparc
Operating System: SunOS 4.1.3
Hostname and Port: Speckle: 8000
Software: C, Gnu C++, Sun C++, and X-Window
Communication: TCP/IP

APPENDIX B - The Movie Application

The sample application schema used for the test scripts is from the data collected by Dr. Gio Weiderhold called the "Movie" database (see diagram).



APPENDIX C - Open OODB Test Programs and Results

The testing programs for The Open OODB system are included here in this Appendix C. Although care has been taken to make this Appendix as readable and as complete as possible, some of the programs listed here are not self contained and require other programs in the libraries to exist. The values of the sample database are factitious. Not all results can be shown and these are explained at the end of each test. Some tests are executed interactively, for example the 4 test programs on concurrency control and the outcome of the test are described.

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*      File Name:      pschema1.cc                                      */
/*      Date     :      05/31/94                                         */
/*      Modifing :      06/26/94                                         */
/*      Version  :      1.2                                              */
/*                                                                       */
/*      Test ID No. : BDBF-001                                           */
/*      Description : Create a schema with classes which has attributes. */
/*      Requirements:  A schema could consist of one or more attributes. */
/*      Expected Results:   A class with attributes is created.          */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              persist("my_object")                                     */
/*              persist("my_oid")                                        */
/*                                                                       */
/*      Testing Results : OK. The class PEOPLE with attributes has been  */
/*                            created                                    */
/*                                                                       */
/*************************************************************************/

#ifdef _SCHEMA_H
#else
#define _SCHEMA_H

#define  maxFieldNum  10
#define  maxFieldWidth 64
#define  maxObjNum    56
#define  startOidNum  1000

#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>

typedef char*        s_String;
typedef s_String     producer_type;
typedef s_String     process;
typedef s_String     role_type;
typedef s_String     year;
typedef s_String     gender;
typedef s_String  geography;
typedef s_String  movie_category;
typedef char*        text;

// 10 attributes

class  People
{
protected:
  s_String    id_name;
  s_String      pcode;
  s_String      dcode;
  s_String    dow_start_end;
  s_String     last_name;
  s_String    first_name;
  year            dob;
  year            dod;
  geography    origin;
```

```
  s_String      notes;

public:
  People(s_String, s_String, s_String,  s_String,s_String, s_String,  year, year,
geography, s_String);
  People(){};
  ~People();
  int  create(s_String*);
  int  show();
  void update_value(char*);
  char*  get_id_name();
  char*  get_pcode();
  char*  get_dob();
  char*  get_dod();
  char*  get_origin();

};
#endif
```

```
*****************************
*  Result 001               *
*****************************
```

Please see Result 012.
All of the manipulations are based on the created schema.

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      pschema2.cc                                        */
/*    Date     :      05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-002                                             */
/*    Description : Create a schema with two classes.                    */
/*    Requirements:  One class is a sub-class of the other class.        */
/*    Expected Results:   A class and a sub-class are created.           */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              persist("my_object")                                     */
/*              persist("my_oid")                                        */
/*                                                                       */
/*      Testing Results : OK. The class PEOPLE and the sub-class ACTOR   */
/*                        has been created                               */
/*                                                                       */
/*************************************************************************/

#ifdef _SCHEMA_H
#else
#define _SCHEMA_H

#define  maxFieldNum  10
#define  maxFieldWidth 64
#define  maxObjNum    56
#define  startOidNum  1000

#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>

typedef char*        s_String;
typedef s_String     producer_type;
typedef s_String     process;
typedef s_String     role_type;
typedef s_String     year;
typedef s_String     gender;
typedef s_String  geography;
typedef s_String  movie_category;
typedef char*        text;

// 10 attributes

class  People
{
protected:
  s_String    id_name;
  s_String      pcode;
  s_String      dcode;
  s_String    dow_start_end;
  s_String     last_name;
  s_String    first_name;
  year            dob;
  year            dod;
  geography    origin;
  s_String      notes;
```

```
public:
  People(s_String, s_String, s_String,  s_String,s_String, s_String,  year, year,
geography, s_String);
  People(){};
  ~People();
  int  create(s_String*);
  int  show();
  void update_value(char*);
  char*  get_id_name();
  char*  get_pcode();
  char*  get_dob();
  char*  get_dod();
  char*  get_origin();

};

// 10 attributes will be inherited from People

class  Actor :  public  People
{
  s_String     stagename;
  gender          sex;
  year         dow_start;
  year           dow_end;
  role_type        roles;
  text             notes;

public:
  Actor(s_String,s_String, s_String, s_String, s_String, s_String, year, year,
      geography, s_String, s_String, gender, year, year, role_type, text);

  ~Actor();
  int  create(s_String*);
  int  _create(s_String*);
  int  show();
  friend Cast;
};

#endif

************************************
*  Result 002                      *
************************************

Please see Result 012.
All of the manipulations are based on the created schema.
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      pschema3.cc                                        */
/*    Date      :     05/31/94                                           */
/*    Modifing  :     06/26/94                                           */
/*    Version   :     1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-003                                             */
/*    Description : Create multiple classes and sub-classes.             */
/*    Requirements:  Multiple classes with attributes and sub-classes    */
/*               with attributes are created.  Attributes from           */
/*               super-classes are inherited.                            */
/*    Expected Results:  The total schema is created.                    */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              persist("my_object")                                     */
/*              persist("my_oid")                                        */
/*                                                                       */
/*      Testing Results : OK. The total MOVIE classes have been created  */
/*                                                                       */
/*************************************************************************/

#ifndef _SCHEMA_H
#define _SCHEMA_H

#define  maxFieldNum  10
#define  maxFieldWidth 64
#define  maxObjNum    56
#define  startOidNum  1000

#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>

typedef char*        s_String;
typedef s_String     producer_type;
typedef s_String     process;
typedef s_String     role_type;
typedef s_String     year;
typedef s_String     gender;
typedef s_String  geography;
typedef s_String  movie_category;
typedef char*        text;

// 10 attributes

class  People
{
protected:
  s_String    id_name;
  s_String      pcode;
  s_String      dcode;
  s_String    dow_start_end;
  s_String     last_name;
  s_String    first_name;
  year            dob;
  year            dod;
  geography    origin;
  s_String      notes;
```

```
public:
  People(s_String, s_String, s_String,  s_String,s_String, s_String,
           year, year, geography, s_String);
  People(){};
  ~People();
  int  create(s_String*);
  void update_value(char*);
  int  show();
  char*  get_id_name();
  char*  get_pcode();
  char*  get_dob();
  char*  get_dod();
  char*  get_origin();
};
// 10 attributes will be inherited from People
class  Actor :  public  People
{
  s_String     stagename;
  gender          sex;
  year         dow_start;
  year           dow_end;
  role_type        roles;
  text             notes;
public:
  Actor(s_String,s_String, s_String, s_String, s_String, s_String, year, year,
      geography, s_String, s_String, gender, year, year, role_type, text);
  Actor(){};
  ~Actor();
  int  create(s_String*);
  int  _create(s_String*);
  int  show();
  friend Cast;
};
class  Cast
{
  s_String      film_id;  // == Movie;
  s_String       title;
  Actor         *actor;
  Director      *director;
  Award         *actor_award;
  role_type       actor_role;
  text           role_desc;
  s_String       notes;
public:
  int  create(s_String*);
  int  show();
  void get_data(s_String*, char*);
};
// 12 attributes of class Movie
class  Movie
{
  s_String           film_id;
  s_String           title;
  year                  ref_year;
  s_String        director;
  s_String        stdios;
  producer_type      producers;
  s_String                dist;
  process                  prc;
  movie_category      category;
  s_String              awards;
  s_String        location;
  text               notes;
public:
    int show();
```

```
    int create(s_String*);
    void get_data(s_String*, char*);
    friend Cast;
    friend Director;
};
class  Director: public virtual People
{
  s_String      stagename;
  s_String      company;
  s_String         notes;
  Movie  *movies;
public :
    int create(s_String*);
    int show();
    int _create(s_String*);
    void get_data(s_String*, char*);
    friend Cast;
};
class  ADirector :  public Actor, public Director
{
  s_String     stagecode;
  s_String     notes;
public:
//  ADirector(s_String , Movie);
    int show();
    int create(s_String*);
    int _create(s_String*);
    void get_data(s_String*, char*);
    friend Cast;
};
class Award
{
  s_String       award_name;
  text               agency;
  s_String            place;
public:
    int show();
    int create(s_String*);
    void get_data(s_String*, char*);
};
#endif

*************************
*  Result 003           *
*************************

Please see Result 012.
All of the manipulations are based on the created schema.
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      pmake_data.cc                                      */
/*    Date      :     05/31/94                                           */
/*    Modifing  :     06/26/94                                           */
/*    Version   :     1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-004                                             */
/*    Description :  Populating data into database with a batch file.    */
/*    Requirements:  Read an external batch file of data and load data   */
/*              into the database with defined schema.                   */
/*    Expected Results:   Loaded data into database and making these     */
/*                    data persistent.                                   */
/*                                                                       */
/*      C++ API commands tested:                                         */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              persist("my_object")                                     */
/*                                                                       */
/*      Testing Results : OK. The 56 records loaded and made persist.    */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>
#include     "schema.h"

static   FILE*  fp;
/*
 * Read data from the  file.
 */
int
createObj(FILE* fp,FILE* fp2, s_String* name)
{
      int     index = 0;
      int     c1,   ch;
      char*   recBuf = new char[maxFieldWidth];
      int     numField ;
      int     recNum = 1 ;
      int         flag ;
      int     count, k, kk;
      char*   movie = new char[maxFieldWidth];
      int      num;
      OODB *p_oodb1 = new OODB("speckle:8000");

      People  *p_people[maxObjNum];
      int  len = 0;
      int  blank;
      flag = 1;
      numField = 0;
      for ( len=0; len<maxObjNum; len++) p_people[len] = new People;
      p_oodb1 -> beginT();
         while (((ch = getc(fp)) != EOF )&&(ch != '#')){};
         fprintf(fp2, "\nRecord No. %d \n", recNum++);
                  num = 0;
           while (ch != EOF ){
                  recBuf[0] = '\0';
```

```
                    count = 0;
                    len = strlen(recBuf);
                    index = 0;
                    movie[0] = '\0';
                    while ((ch != '#' )&&(ch !='\n')){
                       recBuf[index] = ch;
                       flag = 0;
                       ch = getc(fp);
                       index++;
                       };
                    if (flag == 0) {
                    len = strlen(recBuf);
                    recBuf[index] = '\0';
                    len = strlen(recBuf);
                    index++;
                    fprintf(fp2, "%s ", fieldName[numField]);
                    fprintf(fp2, "%s ", recBuf);
                    if (numField < maxFieldNum - 1) {
                    while(recBuf[count] == ' '){count++;};
                    kk = 0;
                    for(k = count ; k <len; k++){
                          if ( recBuf[k] != ' ') {
                          recBuf[kk++] = recBuf[k];
                          };
                    };
                    recBuf[kk]='\0';
                    };
                    strcpy(name[numField], recBuf);
                    numField++;
                    } else   flag = 0;
              if (ch =='#') {
                  c1 = getc(fp);
                  ch = c1;
                  if (c1 =='\n') {
                    numField = 0;
                    flag = 1;
                    movie[0]='\0';
                  p_people[num]->create(name);
                  strcpy(movie,  genOid());
                  p_people[num]->persist(movie);
                  p_people[num]->show();
                  p_people[num]->persist(name[0]);
                 if (num  == 0){
                    printf("Hit any key to continue");
                    blank = getchar( );
                 };
                  num++;
                  fprintf(fp2, "\nRecord No. %d \n", recNum++);
                  while (((ch = getc(fp)) != EOF )&&(ch != '#')){};
                  };
              } else  ch = getc(fp);
          };
       p_oodb1 -> commitT();
       return(-1);
}
```

```
*********************************
*  Result 004                   *
*********************************
```

Please see result 005 and 006 with 56 records stored.
All of the manipulations are based on these data.

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      pfetch1.cc                                         */
/*    Date     :      05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-005                                             */
/*    Description : Retrieving objects using record keys.                */
/*    Requirements:  One or more objects retrieved based upon keys.      */
/*    Expected Results:   Matched records retrieved and shown on screen. */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              fetch("my_object")                                       */
/*              abortT()                                                 */
/*                                                                       */
/*      Testing Results : OK. The objects of PEOPLE with specific names  */
/*                        are retrieved.                                 */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>
#include     "schema.h"

OODB *p_oodb1 = new OODB("speckle:8000");

retrieve()
{
    int       number;
    char     answer;
    char*    key = new char[maxFieldWidth];
    People *p_people = new People;

    printf("==Following is going to retrieve the data from people DB==\n\n\n");
    printf("==Name of the classes is People  \n");
    number = 0;
    p_oodb1 -> beginT();
    printf("Do you want to retrieve OBJECTS? (y/n) \n");
    cin >> answer;
    key[0] = '\0';

    while ( answer == 'y') {
    printf("Do you want to retrieve specific OBJECTS? (y/n) \n");
    cin >> answer;
       while(answer == 'y'){
         printf("Please, give the key: NAME of the PEOPLE\n");
         cin >> key    ;
         if ((p_people = (People*) p_oodb1->fetch(key)) != NULL){
         p_people -> show();
         } else ("The NAME: %s record is not exisiting in the DB.\n", key);
         printf("Do you want to retrieve  more OBJECTS? (y/n) \n");
         cin >> answer;
       };
    printf("Do you want to quit the session(y/n)? \n");
```

```
      cin >> answer;
    if (answer == 'y') answer = 'n';
    else { printf("\n\n------------------------\n\n");
         answer = 'y';
    };
    };
    printf("\n\n Successfully!\n\n");
    p_oodb1 -> abortT();
    return(0);
}

*****************************
*  Result 005               *
*****************************

> Fetch_people_key
   Hurwitz
==Following is going to retrieve the data from people DB==

==Name of the classes is People
      Display starting

id_name :    Hurwitz
pcode      :    871
dcode      :    TAB
Dow_start_end  :    @1977
lastnm     :    Abduladze
firstnm    :   Tengiz
dob        :
dod        :    UN
origin     :    Ru
notes      :    Or(Georgian)

 Successfully!
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OODB                   */
/*                                                                       */
/*    File Name:      pfetch2.cc                                         */
/*    Date      :     05/31/94                                           */
/*    Modifing  :     06/26/94                                           */
/*    Version   :     1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-006                                             */
/*    Description : Retrieve objects using user-generated sequence       */
/*                  number.                                              */
/*    Requirements:  One or more objects retrieved starting at the ith   */
/*                   sequence number.                                    */
/*    Expected Results: The ith objects are retrieved and shown on screen.*/
/*                                                                       */
/*      C++ API commands used:                                           */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              fetch("sequence_No")                                     */
/*              abortT()                                                 */
/*                                                                       */
/*      Testing Results : OK. The designed number of objects starting    */
/*                        with specific sequence number are retrieved.   */
/*                                                                       */
/*************************************************************************/
#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>
#include     "schema.h"

browse()
{
    s_String*  name;
    int           i;
    int      num, number;
    int      start_num;
    int      total;
    char     answer;
    int       blank;
    char*    key = new char[30];
    char*    movie = new char[40];

    printf("==Following is going to retrieve the data from movie DB==\n\n\n");
    printf("==Name of the class is People\n");
    number = 0;
    OODB *p_oodb1 = new OODB("speckle:8000");
    name  = new s_String[12];
    People  *p_people;
    for ( i=0; i<=11; i++) name[i] = new char[30];
    p_oodb1 -> beginT();
          movie[0] = '\0';
          number = 1;
          strcpy(movie,  genOid(1000 + number));
          while ((p_people = (People*) p_oodb1->fetch(movie)) != NULL) {
            number++;
            strcpy(movie,  genOid(1000 + number));
          };
          total = number-1;
          num = number;
```

```
        start_num = 1;
    printf("Do you want to retrieve OBJECTS? (y/n) \n");
    cin >> answer ;
    key[0] = '\0';
    while ( answer == 'y') {
        printf("\nThere are \"%d\" objects \n", total  );
        printf("\nDo you want them all?(y/n)    \n");
        cin >> answer;
        if ( answer != 'y') {
        printf("==How many objects do you want to retrieve? \n");
        cin >> num ;
        cout << "==Starting Object No.:   ";
        cin >> start_num;
        cout << "\n" << flush;
        num = num +  start_num;
        };
        for ( number = start_num; number < num ; number++){
            movie[0] = '\0';
            strcpy(movie,  genOid(number + 1000));
        if ((p_people = (People*) p_oodb1->fetch(movie)) != NULL) {
        printf("\n Object NO.  %d \n", number);
        p_people -> show();
        };

        printf("Hit any key to continue");
        blank = getchar( );
        };     /*  end of FOR  */
       cout<<  "Do you want to quit the session(y/n)? \n" ;
       cin >> answer;
       if (answer == 'y') answer = 'n';
       else { printf("\n\n\n\n");
            answer = 'y';
       };
    };
        p_oodb1 -> abortT();
}
```

```
**************************************
*  Result 006                        *
**************************************

==Following is going to retrieve the data from movie DB==
==Name of the class is People
Do you want to retrieve OBJECTS? (y/n)
 y
There are "56" objects
Do you want them all?(y/n)
 n
==How many objects do you want to retrieve?
  1
==Starting Object No.":  1

 Object NO.  1
id_name :     Aaron
pcode      :    D
dcode      :    PAa
Dow_start_end  :    @1979
lastnm    :     Aaron
firstnm   :    Paul
dob       :    [1]
dod       :    UN
origin    :    Am
notes     :
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*      File Name:      fetch_gid.cc                                     */
/*      Date      :     05/31/94                                         */
/*      Modifing  :     06/26/94                                         */
/*      Version   :     1.2                                              */
/*                                                                       */
/*    Test ID No.: BDBF-007                                              */
/*    Description: Retreive system-assigned Object Identifier (OID).     */
/*    Requirements: OIDs are assigned to objects stored as persistent data*/
/*    Expected Results:  The OID is fetched and reported.                */
/*                                                                       */
/*      C++ API commands tested:                                         */
/*                    ObjectGID(char* name)                              */
/*                    fetch(GID *gid)                                    */
/*                                                                       */
/*      Testing Results : OK. OID retrieved and verified as having the   */
/*                        given key.                                     */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>

#include     "schema.h"

fetch_gid()
{
    s_String*  name;
    int           i;
    int       number;
    int       total;
    int       blank;
    char*     key = new char[30];
    char*     movie = new char[40];

    GID *pgid;
    printf("==Following is going to retrieve the data from movie DB==\n\n\n");
    printf("==Name of the classes is People\n");
    number = 0;
    OODB *p_oodb = new OODB("speckle:8000");
    name  = new s_String[12];
    People  *p_people = new People;
    for ( i=0; i<=11; i++) name[i] = new char[30];
    p_oodb -> beginT();
          movie[0] = '\0';
          number = 1;
          strcpy(movie,  genOid(1000 + number));

            //   Following is for testing the GID

          while ((pgid=  p_oodb->ObjectGID(movie)) != NULL) {
            if ((p_people = (People*) p_oodb->fetch(pgid)) != NULL) {
            printf("\n Object NO.  %d \n", number);
            p_people -> show();
            printf("Hit any key to continue");
            blank = getchar( );
            };
```

```
            cout <<"\n Object ID %s  ";
            cout <<  pgid;
            number++;
            strcpy(movie,  genOid(1000 + number));
          };
            cout <<"\n";
          total = number-1;
          printf("\nThere are \"%d\" objects \n", total  );
           p_oodb -> abortT();

}

*****************************
*  Result 007               *
*****************************

==Following is going to retrieve the data from movie DB==

==Name of the classes is People

 Object NO.  1

 Object ID is 60fa78


id_name :    Aaron
pcode      :    D
dcode      :    PAa
Dow_start_end  :    @1979
lastnm    :    Aaron
firstnm   :   Paul
dob       :    1923
dod       :    1966
origin    :    American
notes     :

Object NO. 2
 . . .
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                   for                                 */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      query_set.cc                                       */
/*    Date      :     05/31/94                                           */
/*    Modifing  :     06/26/94                                           */
/*    Version   :     1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-009                                             */
/*    Description : Create a set of a class.                             */
/*    Requirements:  Create a collection of objects: set or list.  This  */
/*             is necessary for executing Object Query Language.         */
/*    Expected Results:   A set of class is created.                     */
/*                                                                       */
/*      C++ API commands used:                                           */
/*           OQL                                                         */
/*           Set<myclass>                                                */
/*           Set_myclass *set                                            */
/*           set->InsertMember()                                         */
/*               set->persist("name_of_my_class_set")                    */
/*               populating 56 objects into the SET                      */
/*               OODB *my_oodb(hostname:port)                            */
/*               beginTransaction()                                      */
/*               commitTransaction()                                     */
/*               persist("my_object")                                    */
/*               persist("my_oid")                                       */
/*                                                                       */
/*      Testing Results : OK. The set of class people created            */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>
#include     "schema.h"
#include     <Iterator.h>
#include     "List.h"
#include     "Set.h"
DECLARE      Set<People>
IMPLEMENT    Set<People>
IMPLEMENT    List<People>

createSetObj(FILE* fp,FILE* fp2, s_String* name)
{
    OODB *p_oodb1 = new OODB("speckle:8000");
    char*   people = new char[40];

      int count, k2, kk;
        int     index = 0;
      int     c1,   ch;
      char*   recBuf = new char[maxFieldWidth];
      int     numField ;
      int     recNum = 1 ;
      int         flag ;
      int      num;

      char*   movie = new char[maxFieldWidth];
      People  *p_people[maxObjNum];
      int  len = 0;
```

```
        flag = 1;
        numField = 0;
        for ( len=0; len<maxObjNum; len++) p_people[len] = new People;

        p_oodb1 -> beginT();
        Set_People *set = new Set_People();
          printf( "\n|Set_of_people  | = %d\n", set->Cardinality() );
        Iterator k;
        People  l;
        for(set->First_Member(k); !set->Is_End_Of_Set(k); set->Next_Member(k)) {
              l  =  set->Get_Member(k);
              l.show();
        }

             while (((ch = getc(fp)) != EOF )&&(ch != '#')){};
           fprintf(fp2, "\nRecord No. %d \n", recNum++);
           num = 0;
//         count = 0;
//           while ((ch != EOF )|| (count < 56)){
             while (ch != EOF ){
                     recBuf[0] = '\0';
                     len = strlen(recBuf);
                     index = 0;
                     movie[0] = '\0';

                     while ((ch != '#' )&&(ch !='\n')){
                        recBuf[index] = ch;
                        flag = 0;
                        ch = getc(fp);
                        index++;
                        };
                     if (flag == 0) {
                       len = strlen(recBuf);
                     recBuf[index] = '\0';
                       len = strlen(recBuf);
                     index++;
                     fprintf(fp2, "%s ", fieldName[numField]);
                     fprintf(fp2, "%s ", recBuf);

                     count = 0;
                           if (numField < maxFieldNum - 1) {
                          while(recBuf[count] == ' '){count++;};
                          kk = 0;
                          for(k2 = count ; k2 <len; k2++){
                                  if ( recBuf[k2] != ' ') {
                                  recBuf[kk++] = recBuf[k2];
                                  };
                          };
                          recBuf[kk]='\0';
                          };

                     strcpy(name[numField], recBuf);
                     numField++;
                     } else   flag = 0;
               if (ch =='#') {
                   c1 = getc(fp);
                   ch = c1;
                   if (c1 =='\n') {
                       numField = 0;
                       flag = 1;
                       movie[0]='\0';
                   p_people[num]->create(name);
                   p_people[num]->persist(name[0]);
    /* Can't just use p->per.., set->Insert_Member(p), Set can not tell p1, p2..
*/
```

```
                set->Insert_Member(*p_people[num]);
                p_people[num]->show();
                num++;

                fprintf(fp2, "\nRecord No. %d \n", recNum++);
                      while (((ch = getc(fp)) != EOF )&&(ch != '#')){};
                 };
            } else  ch = getc(fp);
//          count++;
         };
      set->persist("Set_of_People");
      p_oodb1 -> commitT();
//    p_oodb1 -> abortT();
        return(-1);
}

************************************
*  Result 009                      *
************************************

Please see result 010.
The creation of a set is used for the Object Query Language.
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
*/                                                                        */
/*                                                                        */
/*    File Name:      query_eql.cc                                        */
/*    Date      :     05/31/94                                            */
/*    Modifing  :     06/26/94                                            */
/*    Version   :     1.2                                                 */
/*                                                                        */
/*    Test ID No. : BDBF-010.1                                            */
/*    Description : Select objects using OQL where COMPARE_EQUAL          */
/*    Requirements: Select-From-Where statement with equality.            */
/*    Expected Results:   Matched results are reported.                   */
/*                                                                        */
/*      C++ API commands used:                                            */
/*              OODB *my_oodb(hostname:port)                              */
/*              beginTransaction()                                        */
/*              commitTransaction()                                       */
/*              persist("my_object")                                      */
/*              persist("my_oid")                                         */
/*           Query{                                                      */
/*                result = SELECT pp                                     */
/*                FROM People pp  IN Set_of_People                       */
/*                WHERE compare_equl( pp.get_id_name(), KEY    ) ;       */
/*             }                                                          */
/*                                                                        */
/*      Testing Results : OK.  Matched results are reported.              */
/*                                                                        */
/*************************************************************************/

#include     "new.h"
#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <strings.h>
#include     <iostream.h>
#include     <Iterator.h>
#include     "List.h"
#include     "Set.h"
#include     "schema.h"

DECLARE      Set<People>
IMPLEMENT    Set<People>
IMPLEMENT    List<People>

query_by_name()
{
    char*    KEY = new char[30];
    Set_People *result ;
    Iterator k;
    People m;

    KEY[0] = '\0';
    printf("You want to find the person's record, NAME equal :  \n");
    cin >>KEY;
    Query{
      result = SELECT pp
             FROM People pp  IN Set_of_People
             WHERE compare_equl( pp.get_id_name(), KEY    ) ;
    }
      /* Print the result of the query */

        printf( "\n\"%s's\" RECORD is :    \n", KEY );
        if (result->Cardinality() != 0) {
```

```
            for( result->First_Member(k); !result->Is_End_Of_Set(k);
                                          result->Next_Member(k) ) {
            m = result->Get_Member(k);
            m.show();
        } };
        printf( "Total Records found = %d\n\n\n\n", result->Cardinality() );
      return 0;
}

queryObj()
{
    s_String*  name;
    int           i;
    int       number;
    char     answer;
    char*    KEY = new char[30];
    Iterator k;

    p_oodb1 -> beginT();
    Set_People *set  = ( Set_People*) OpenOODB->fetch("Set_of_People");

    printf("==Name of the classes is People  \n");
    number = 0;
    name  = new s_String[12];
    People *p_people = new People[1];
    for ( i=0; i<=11; i++) name[i] = new char[30];
    printf("Do you want to retrieve OBJECTS? (y/n) \n");
    cin >> answer;

    while ( answer == 'y') {
    printf("Do you want to retrieve specific OBJECTS? (y/n) \n");
    cin >> answer;
       while(answer == 'y'){
         printf("Please, give the KEY: NAME of the PEOPLE\n");
           cin >> KEY    ;
         if (( p_people = (People*) p_oodb1->fetch(KEY)) != NULL){
         p_people -> show();
         }else ("The NAME: %s record is not exisiting in the DB.\n", KEY);

          printf("Do you want to retrieve  more OBJECTS? (y/n) \n");
         cin >> answer;
      };

    printf("\n\n\n");

    printf("Now we are using OQL to retrieve OBJECTS    === \n");
      browse_all_of_set();
      query_by_name();

    printf("================================================ \n");
    printf("Do you want to quit the session(y/n)? \n");
      cin >> answer;
    if (answer == 'y') answer = 'n';
    else { printf("\n\n-------------------------\n\n");
         answer = 'y';
    };
    };
    printf("\n\n Successfully!\n\n");
    p_oodb1 -> abortT();
    return(0);
}
```

```
**********************************
*  Result 010.1                  *
**********************************
>
You want to find the person's record, NAME equal :
       Yang

      Display starting

      id_name :    Yang
      pcode       :    871
      dcode       :    GS-16
      Dow_start_end  :    @1928-1994
      lastnm     :    Yang
      firstnm    :   John
      dob        :    1901
      dod        :    UN
      origin     :    American
      notes      :    VIK

Hit any key to continue:

Total Records found = 1
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                   for                                 */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      query_les.cc                                       */
/*    Date      :     05/31/94                                           */
/*    Modifing  :     06/26/94                                           */
/*    Version   :     1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-010.2                                           */
/*    Description : Select objects using OQL where LESS.                 */
/*    Requirements:  Select-From-Where statement with inequality.        */
/*    Expected Results:   Less results are reported.                     */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              persist("my_object")                                     */
/*              persist("my_oid")                                        */
/*           Query{                                                      */
/*                result = SELECT pp                                     */
/*                FROM People pp  IN Set_of_People                       */
/*                WHERE compare_less( KEY, pp.get_dod()    ) ;           */
/*            }                                                          */
/*                                                                       */
/*      Testing Results : OK. Correct records are selected.              */
/*                                                                       */
/*************************************************************************/

#include     "new.h"
#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <strings.h>
#include     <iostream.h>
#include     <Iterator.h>
#include     "List.h"
#include     "Set.h"
#include     "schema.h"

DECLARE      Set<People>
IMPLEMENT    Set<People>
IMPLEMENT    List<People>

query_less_dod()
{
    char*    KEY = new char[30];
    Set_People *result ;
    Iterator k;
    People m;

    KEY[0] = '\0';
    printf("in terms of DATE of death   \n", );
    printf("You want to find the person who died before year  \n");
    cin >>KEY;
      Query{
      result = SELECT p_doda
             FROM People p_doda IN Set_of_People
             WHERE compare_less( KEY, p_doda.get_dod() )  ;
    }

        printf( "\n*******Results the persons died before year %s ********\n",
KEY );
        if (result->Cardinality() != 0) {
```

```
            for( result->First_Member(k); !result->Is_End_Of_Set(k);
                                          result->Next_Member(k) ) {
            m = result->Get_Member(k);
            m.show();
        printf("Hit any key to continue:");
      getchar();
      } };
        printf( "\nTotal records found = %d\n", result->Cardinality() );
      return 0;
}
      /**  Following is the query by origin....      ***/

queryObj()
{
    s_String*  name;
    int           i;
    int       number;
    char     answer;
    char*    KEY = new char[30];
    Iterator k;

    p_oodb1 -> beginT();
    Set_People *set  = ( Set_People*) OpenOODB->fetch("Set_of_People");

    printf("==Name of the classes is People  \n");
    number = 0;
    name  = new s_String[12];
    People *p_people = new People[1];
    for ( i=0; i<=11; i++) name[i] = new char[30];
    printf("Do you want to retrieve OBJECTS? (y/n) \n");
    cin >> answer;

    while ( answer == 'y') {
    printf("Do you want to retrieve specific OBJECTS? (y/n) \n");
    cin >> answer;
       while(answer == 'y'){
         printf("Please, give the KEY: NAME of the PEOPLE\n");
           cin >> KEY    ;
         if (( p_people = (People*) p_oodb1->fetch(KEY)) != NULL){
         p_people -> show();
         }else ("The NAME: %s record does not exist in the DB.\n", KEY);

           printf("Do you want to retrieve  more OBJECTS? (y/n) \n");
         cin >> answer;
      };

    printf("\n\n\n");

    printf("Now we are using OQL to retrieve OBJECTS    === \n");
      query_less_dod();
    printf("============================================== \n");
    printf("Do you want to quit the session(y/n)? \n");
      cin >> answer;
    if (answer == 'y') answer = 'n';
    else { printf("\n\n-------------------------\n\n");
         answer = 'y';
    };
    };
    printf("\n\n Successfully!\n\n");
    p_oodb1 -> abortT();
    return(0);
}
```

```
*******************************
*  Result 010.2               *
*******************************

You want to find the person who died before year
    1995

*******Results the persons died before year 1995 ********
      Display starting

id_name :    Aaron
pcode       :    D
dcode       :    PAa
Dow_start_end  :    @1979
lastnm     :    Aaron
firstnm    :   Paul
dob        :    [1]
dod        :    1994
origin     :    Am
notes      :

Hit any key to continue:

Total Records found = 1
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Teax Ins. OOODB                          */
/*                                                                       */
/*    File Name:      query_gre.cc                                       */
/*    Date      :     05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-010.3                                           */
/*    Description : Select objects using OQL where GREATER.              */
/*    Requirements:  Select-From-Where statement with inequality.        */
/*    Expected Results:   Greater results are reported.                  */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              persist("my_object")                                     */
/*              persist("my_oid")                                        */
/*           Query{                                                      */
/*                result = SELECT pp                                     */
/*                FROM People pp  IN Set_of_People                       */
/*                WHERE compare_greater( KEY, pp.get_dob()    ) ;        */
/*                }                                                      */
/*                                                                       */
/*      Testing Results : OK.  Correct records are selected.             */
/*                                                                       */
/*************************************************************************/

#include     "new.h"
#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <strings.h>
#include     <iostream.h>
#include     <Iterator.h>
#include     "List.h"
#include     "Set.h"
#include     "schema.h"

DECLARE      Set<People>
IMPLEMENT    Set<People>
IMPLEMENT    List<People>

query_greater_dob()
{
    char*    KEY = new char[30];
    Set_People *result ;
    Iterator k;
    People m;

    KEY[0] = '\0';
    printf("in terms of DATE of birth   \n", );
    printf("You want to find the person who were born after the year of  \n");

    cin >>KEY;
      Query{
      result = SELECT p_doba
             FROM People p_doba IN Set_of_People
             WHERE compare_greater( p_doba.get_dob(), KEY )  ;
    }


       printf( "\n*******Results the person were born after the year %s
```

```
********\n", KEY );

        if (result->Cardinality() != 0) {
             for( result->First_Member(k); !result->Is_End_Of_Set(k);
                                          result->Next_Member(k) ) {
            m = result->Get_Member(k);
            m.show();
        printf("Hit any key to continue:");
       getchar();
       } };
        printf( "Total records found  = %d\n", result->Cardinality() );
       return 0;
}

queryObj()
{
    s_String*  name;
    int           i;
    int       number;
    char     answer;
    char*    KEY = new char[30];
    Iterator k;

    p_oodb1 -> beginT();
    Set_People *set  = ( Set_People*) OpenOODB->fetch("Set_of_People");

    printf("==Name of the classes is People  \n");
    number = 0;
    name  = new s_String[12];
    People *p_people = new People[1];
    for ( i=0; i<=11; i++) name[i] = new char[30];
    printf("Do you want to retrieve OBJECTS? (y/n) \n");
    cin >> answer;

    while ( answer == 'y') {
    printf("Do you want to retrieve specific OBJECTS? (y/n) \n");
    cin >> answer;
       while(answer == 'y'){
         printf("Please, give the KEY: NAME of the PEOPLE\n");
           cin >> KEY    ;
         if (( p_people = (People*) p_oodb1->fetch(KEY)) != NULL){
         p_people -> show();
         }else ("The NAME: %s record is not exisiting in the DB.\n", KEY);

           printf("Do you want to retrieve  more OBJECTS? (y/n) \n");
         cin >> answer;
      };

    printf("\n\n\n");

    printf("Now we are using OQL to retrieve OBJECTS    === \n");
      query_greater_dob();

    printf("============================================== \n");
    printf("Do you want to quit the session(y/n)? \n");
      cin >> answer;
    if (answer == 'y') answer = 'n';
    else { printf("\n\n-------------------------\n\n");
         answer = 'y';
    };
    };
    printf("\n\n Successfully!\n\n");
    p_oodb1 -> abortT();
    return(0);
}
```

```
*****************************
*  Result 010.3             *
*****************************

In terms of DATE of birth
You want to find the person who were born after year
   1945

*******Results the persons were born after the year 1945 ********
      Display starting

id_name :    Zhang
pcode      :    CD
dcode      :    YMZ
Dow_start_end  :    @1980-1992
lastnm     :    Zhang
firstnm    :   YiMou
dob        :    1950
dod        :    UN
origin     :    American
notes      :     Lw(Gong Li)

Hit any key to continue:

      Display starting

id_name :    Yule
pcode      :    D
dcode      :    PYu
Dow_start_end  :    @1993
lastnm     :    Yule
firstnm    :   Paul
dob        :    1954
dod        :    UN
origin     :    Br
notes      :     Ty(TV)

Total Records found = 2
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                   for                                 */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*      File Name:      query.cc                                         */
/*      Date     :      05/31/94                                         */
/*      Modifing :      06/26/94                                         */
/*      Version  :      1.2                                              */
/*                                                                       */
/*      Test ID No. : BDBF-010.4                                         */
/*      Description : Retrieving objects by OQL with AND.                */
/*      Requirements: Select-From-Where statement with conjunction.      */
/*      Expected Results : Matched results are reported.                 */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              persist("my_object")                                     */
/*              persist("my_oid")                                        */
/*            Query{                                                     */
/*                 result = SELECT pp                                    */
/*                 FROM People pp  IN Set_of_People                      */
/*                 WHERE compare_equl( KEY, pp.get_dob() ) &&            */
/*                        compare_equl(KEY1, pp.get_origin())            */
/*             }                                                        */
/*                                                                       */
/*      Testing Results : OK.  Matched results are reported.             */
/*                                                                       */
/*************************************************************************/

#include     "new.h"
#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <strings.h>
#include     <iostream.h>
#include     <Iterator.h>
#include     "List.h"
#include     "Set.h"
#include     "schema.h"

DECLARE      Set<People>
IMPLEMENT    Set<People>
IMPLEMENT    List<People>

query_equl_dob_and_origin()
{
    char*    KEY = new char[30];
    char*    KEY1 = new char[30];
    Set_People *result;
    Iterator k;
    People m;

    KEY[0] = '\0';
    printf("\n\n", );
    printf("You want to find the person who were born at the year of  \n");
    cin >>KEY;
    printf("And his/her origin is \n");
    cin >>KEY1;
        Query{
        result = SELECT p_doby
                 FROM People p_doby IN Set_of_People
                 WHERE compare_equl( KEY, p_doby.get_dob() ) &&
                   compare_equl(KEY2, p_doby.get_origin())
```

```
    }
        printf( "\n*******Results the person were born at the year %s
********\n", KEY );
        printf( "\n*******And his/her origin is %s ********\n", KEY1 );
        if (result->Cardinality() != 0) {
                for( result->First_Member(k); !result->Is_End_Of_Set(k);
                                        result->Next_Member(k) ) {
            m = result->Get_Member(k);
            m.show();
        printf("Hit any key to continue:");
        getchar();
        } };
        printf( "Total records found = %d\n", result->Cardinality() );
        return 0;
}

************************************
*  Results 10.4                    *
************************************

You want to find the person who were born at the year of
  1924
And his/her origin is
  Po

*******Results the person were born at the year 1924 ********
*******And his/her origin is Po
      Display starting

id_name :    Zanussi
pcode      :    D
dcode      :    KZs
Dow_start_end  :    @1985
lastnm     :    Zanussi
firstnm    :   Krystof
dob        :    1924
dod        :    1995
origin     :    Po
notes      :
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                   for                                 */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      query_or.cc                                        */
/*    Date      :     05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-010.5                                           */
/*    Description : Select objects using OQL qualified with OR.          */
/*    Requirements:  Select-From-Where statement with disjunction.       */
/*    Expected Results:  Matched results are reported.                   */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              persist("my_object")                                     */
/*              persist("my_oid")                                        */
/*           Query{                                                      */
/*                result = SELECT pp                                     */
/*                FROM People pp  IN Set_of_People                       */
/*                WHERE compare_equl( KEY, pp.get_dob() ) OR             */
/*                       compare_equl(KEY1, pp.get_origin())             */
/*            }                                                          */
/*                                                                       */
/*      Testing Results : OK.  Correct results are reported.             */
/*                                                                       */
/*************************************************************************/

#include     "new.h"
#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <strings.h>
#include     <iostream.h>
#include     <Iterator.h>
#include     "List.h"
#include     "Set.h"
#include     "schema.h"
DECLARE      Set<People>
IMPLEMENT    Set<People>
IMPLEMENT    List<People>
query_less_greater_dob()
{
    char*    KEY = new char[30];
    char*    KEY1 = new char[30];
    Set_People *result;
    Iterator k;
    People m;
    KEY[0] = '\0';
    printf("\n\n", );
    printf("You want to find the person who were born at the year of  \n");
    cin >>KEY;
    printf("OR his/her origin is  \n");
    cin >>KEY1;
        Query{
        result = SELECT p_doby
                 FROM People p_doby IN Set_of_People
                 WHERE compare_equl( KEY, p_doby.get_dob() ) OR
                    compare_equl(KEY1, p_doby.get_dob()) ;
    }
        printf( "\n*******Results the person were born at the year %s
********\n", KEY );
```

```
        printf( "\n*******Or his/her origin is \n", KEY1 );
        if (result->Cardinality() != 0) {
                for( result->First_Member(k); !result->Is_End_Of_Set(k);
                                              result->Next_Member(k) ) {
            m = result->Get_Member(k);
            m.show();
        printf("Hit any key to continue:");
        getchar();
        } };
        printf( "Total records found = %d\n", result->Cardinality() );
        return 0;
}

****************************************
*  Result 10.05                        *
****************************************

You want to find the person who were born at the year of
  1915
Or  his/her origin is
  Br
*******Results the person were born at the year 1915 ********
*******Or his/her origin is Br
      Display starting
id_name :    T.Young
pcode      :    D
dcode      :    TYg
Dow_start_end  :    @1948-1980
lastnm    :    Young
firstnm   :   Terence
dob       :    1915
dod       :    1990
origin    :    Br
notes     :

Hit any key to continue:
      Display starting
id_name :    Yule
pcode      :    D
dcode      :    PYu
Dow_start_end  :    @1993
lastnm    :    Yule
firstnm   :   Paul
dob       :    1916
dod       :    1991
origin    :    Br
notes     :     Ty(TV)

Hit any key to continue:
      Display starting
id_name :    Asquith
pcode      :    D
dcode      :    AA
Dow_start_end  :    1928
lastnm    :    Asquith
firstnm   :   Anthony'Puffin'
dob       :    1902
dod       :    1968
origin    :    Br
notes     :

Total records found = 3
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      update.cc                                          */
/*    Date      :     05/31/94                                           */
/*    Modifing  :     06/26/94                                           */
/*    Version   :     1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-011                                             */
/*    Description : Read, modify and write objects.                      */
/*    Requirements:  Known objects to be read, its values can be         */
/*                   modified and write back to persistent store.        */
/*    Expected Results: Objects can be updated.                          */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              fetch("my_object")                                       */
/*              persist("my_object")                                     */
/*              persist("my_oid")                                        */
/*                                                                       */
/*      Testing Results : OK.  Object read, updated, and stored back.    */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>
#include     "schema.h"

OODB *p_oodb1 = new OODB("speckle:8000");

void People::update_value(char *key)
{
    char value[maxFieldWidth];
    printf("\n\n");
    printf("Editing the Value      \n");
    printf("Original Record Key is \n\n");
    printf("id_name :    %s\n\n", id_name);
    strcpy(key, id_name);
    printf("Updating the record ----------\n\n\n");
    printf("Field ( 1): pcode      :    %s\n", pcode     );
    printf("Input New Value        :    \n" );
    scanf(" %s", pcode  );
    printf("\nField ( 2): dcode      :    %s\n", dcode     );
    printf("Input New Value        :    \n" );
    scanf(" %s", dcode  );
    printf("\nField ( 3): Dow_start_end  :    %s\n", dow_start_end );
    printf("Input New Value        :    \n" );
    scanf(" %s", dow_start_end  );
    printf("\nField ( 4): last_name      :    %s\n", last_name   );
    printf("Input New Value        :    \n" );
    scanf(" %s", last_name  );
    printf("\nField ( 5): first_name      :    %s\n", first_name   );
    printf("Input New Value        :    \n" );
    scanf(" %s", first_name  );
    printf("\nField ( 6): dob      :    %s\n", dob   );
    printf("Input New Value        :    \n" );
```

```
    scanf(" %s", dob  );
    printf("\nField ( 7): dod      :    %s\n", dod   );
    printf("Input New Value       :    \n" );
    scanf(" %s", dod  );
    printf("\nField ( 8): origin   :    %s\n", origin   );
    printf("Input New Value       :    \n" );
    scanf(" %s", origin );
    printf("\nField ( 9): notes    :    %s\n", notes    );
    printf("Input New Value       :    \n" );
    scanf(" %s", notes );
}

update_by_key()
{
    int       number;
    char     answer;
    char*    key = new char[maxFieldWidth];
    char*    people  = new char[maxFieldWidth];
    People *p_people = new People;

    printf("==Following is going to retrieve the data from people DB==\n\n\n");
    printf("==Name of the classes is People  \n");
    number = 0;
    p_oodb1 -> beginT();
    printf("Do you want to retrieve and update  OBJECTS? (y/n) \n");
    cin >> answer;
    key[0] = '\0';

       while(answer == 'y'){
         printf("Please, give the key: NAME of the PEOPLE\n");
         cin >> key    ;
         if ((p_people = (People*) p_oodb1->fetch(key)) != NULL){
         p_people -> show();
         p_people -> update_value(people);
         p_people -> show();
         } else ("The NAME: %s record is not exisiting in the DB.\n", key);
         printf("Are you sure all of the values are right?(y/n) \n");
         cin >> answer;
         if ( answer == 'y') p_people->persist(people); else printf(" You can
modify it later on !!\n");
         printf("Do you want to retrieve  more OBJECTS? (y/n) \n");
         cin >> answer;
      };
   p_oodb1 -> commitT();
}

**********************************
*  Result 011                    *
**********************************

==Following is going to retrieve the data from people DB==

==Name of the classes is People
Do you want to retrieve and update  OBJECTS? (y/n)
 Y
Please, give the key: NAME of the PEOPLE
 Yang
 <pre><h3>       Display starting

id_name :    Yang
pcode      :    871
dcode      :    TAb
Dow_start_end  :    @1977
lastnm    :    Abduladze
firstnm   :   Tengiz
```

```
dob        :
dod        :    UN
origin     :    Ru
notes      :    Or(Georgian)

Editing the Value
Original Record Key is

id_name :    Yang

Updating the record ----------

Field ( 1): pcode       :    871
Input New Value         :
871

Field ( 2): dcode       :    TAb
Input New Value         :
GS-16

Field ( 3): Dow_start_end  :    @1977
Input New Value         :
@1928-1994

Field ( 4): last_name       :    Abduladze
Input New Value         :
Yang

Field ( 5): first_name      :    Tengiz
Input New Value         :
Tengiz

Field ( 6): dob      :
Input New Value         :
1900

Field ( 7): dod      :    UN
Input New Value         :
UN

Field ( 8): origin   :    Ru
Input New Value         :
American

Field ( 9): notes    :    Or(Georgian)
Input New Value         :
Very Important
 <pre><h3>       Display starting

id_name :    Yang
pcode       :    871
dcode       :    GS-16
Dow_start_end  :    @1928-1994
lastnm      :    Yang
firstnm     :   Tengiz
dob         :    1900
dod         :    UN
origin      :    American
notes       :    Very

Do you want to retrieve  more OBJECTS? (y/n)
n

  Successfully!
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      pinfo.cc                                           */
/*    Date     :      05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-012                                             */
/*    Description : Retrieving a specific class from dictionary          */
/*    Requirements: The "Type" information based upon the defined        */
/*                  schema exist in the dictionary.                      */
/*    Expected Results: The type information for a given class is        */
/*                      reported.                                        */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              TYPEINFO_LIST *list = ty_base.BaseList()                 */
/*              MEMBER_LIST *mlist = ty.MemberList()                     */
/*                                                                       */
/*      Testing Results : OK. The name of class and its attributes and   */
/*                        type information are reported.                 */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>

#include     "schema.h"

OODB *p_oodb = new OODB("speckle:8000");

retrieveBaseinfo()
{
    int           i;
    int       number;
    char     answer;
    char*    key = new char[30];

    number = 0;
    People *p_people = new People;
    Actor  *p_actor  = new Actor;
    p_oodb -> beginT();

    TYPEINFO &ty = p_people->oodb_typeof();
    TYPEINFO &ty_base = p_actor->oodb_typeof();
    MEMBER_LIST *mlist = ty.MemberList();
    if (mlist == NULL ) printf(" List is NULL!!\n");

    for(; mlist !=NULL; mlist = mlist->Next())
        printf("%s: %s\n", mlist->Name(), mlist->Type());

    printf("\n\n Successfully!\n\n");
    p_oodb -> abortT();
    return(0);
}
```

```
****************************
*  Result 012              *
****************************


==Name of the classes is People
{Type Poeple:  64 bytes, defined in file ./schema.h at line 51
Members of class People:
id_name: t8s_String
pcode: t8s_String
dcode: t8s_String
dow_start_end: t8s_String
last_name: t8s_String
first_name: t8s_String
dob: t4year
dod: t4year
. . .
}
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      dictionary.cc                                      */
/*    Date     :      05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-013                                             */
/*    Description :  Retrieving a class schema and all its sub-class     */
/*                   schema from dictionary.                             */
/*    Requirements:  All of class definition and its sub-class type and  */
/*                   attributes are reported.                            */
/*    Expected Results:   The class and sub-classes' name and type       */
/*                        information are reported.                      */
/*                                                                       */
/*      C++ API commands used:                                           */
/*                                                                       */
/*           TYPEINFO &tyinfo = p_people->oodb_typeof();                */
/*           tyinfo.describeall();                                      */
/*                                                                       */
/*      Testing Results : OK.  The class PEOPLE and its sub-classs ACTOR */
/*                        meta information are reported.                 */
/*                                                                       */
/*************************************************************************/
#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>
#include     "schema.h"

OODB *p_oodb = new OODB("speckle:8000");

getinfo()
{
//    s_String*  name;
    int           i;
    int      num, number;
    int      start_num;
    int      total;
    char     answer;
    int       blank;
    char*    key = new char[30];
    char*    movie = new char[40];

    printf("==Following is going to retrieve the data from movie DB==\n\n\n");
    printf("==Name of the classes is People\n");
    People  *p_people = new People;
    p_oodb -> beginT();
    number = 0;
    TYPEINFO &tyinfo = p_people->oodb_typeof();
    cout << "\n ***** tyinfo.describeall();\n ";
    tyinfo.describeall();
          num = number;
          start_num = 1;
    printf("Do you want to retrieve OBJECTS? (y/n) \n");
    cin >> answer ;
    key[0] = '\0';
    while ( answer == 'y') {
          printf("\nThere are \"%d\" objects \n", total  );
```

```
          printf("\nDo you want them all?(y/n)     \n");
          cin >> answer;
          if ( answer != 'y') {
          printf("==How many objects do you want to retrieve? \n");
          cin >> num ;
          cout << "==Starting Object No.:    ";
          cin >> start_num;
          cout << "\n" << flush;
          num = num +  start_num;
          };
          for ( number = start_num; number < num ; number++){
                movie[0] = '\0';
                strcpy(movie,  genOid(number + 1000));
          if ((p_people = (People*) p_oodb->fetch(movie)) != NULL) {
          printf("\n Object NO.  %d \n", number);
          p_people -> show();
          };

          printf("Hit any key to continue");
          blank = getchar( );
          };     /*  end of FOR  */
      cout<<  "Do you want to quit the session(y/n)? \n" ;
       cin >> answer;
    if (answer == 'y') answer = 'n';
    else { printf("\n\n------------------------\n\n");
         answer = 'y';
    };
    };
          p_oodb -> commitT();
    printf("\n\n Successfully!\n\n");
}

************************************
*  Result 013                      *
************************************

Actor is Derived from People.
 *** tyinfo.describeall();

{Type Actor:  88 bytes, defined in file ./schema.h at line 76
Derived from:  public People
  {Type People:  64 bytes, defined in file ./schema.h at line 51
   Members of class people:
     id_name: t8s_String
     pcode: t8s_String
     dcode: t8s_String
     dow_start_end: t8s_String
     last_name: t8s_String
     first_name: t8s_String
     . . .
}
 Members of class Actor:
   Stagename:  t8s_String
   Sex: t6gender
   . . .
}
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*      File Name:      inherit2.cc                                      */
/*      Date     :      05/31/94                                         */
/*      Modifing :      06/26/94                                         */
/*      Version  :      1.2                                              */
/*                                                                       */
/*    Test ID No. : BDBF-015                                             */
/*    Description : Create a class and a sub-class and allow all the     */
/*                   attributes to be inherited to the sub-class.        */
/*    Requirements:  The attributes from the super-class should be       */
/*                  inherited to the sub-class.                          */
/*    Expected Results:   All the attributes are inherited in the sub-   */
/*                        class.                                         */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              persist("my_object")                                     */
/*                                                                       */
/*      Testing Results : Using C++ constructor, this test cannot be     */
?*                        committed.  Failure reported as Bus error.     */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>
#include     "schema.h"

OODB *p_oodb1 = new OODB("speckle:8000");

inheritanceFail()
{
    s_String*  name;
    char*   people = new char[maxFieldWidth];
    char*   actor = new char[maxFieldWidth];
    char*   answer;
    int      num, i;
    name  = new s_String[maxFieldNum];
    answer = new char[3];
    answer[0] = '\0';
    num = 0;
    p_oodb1 -> beginT();

    for ( i=0; i<=maxFieldNum; i++) name[i] = new char[maxFieldWidth];
    printf("Do you want to set up PEOPLE objects?(Y/N)  \n");
    while ((answer[0] = getchar()) == 'y')   {
    get_data(name , people);
       People  p_people(name[0], name[1], name[2], name[3], name[4], name[5],
                        name[6], name[7],  name[8], name[9]);
       p_people.show();
    cout << " Is " << name[0] << " a Actor?(Y/N)";
    cin >> answer;
    if ( answer[0] == 'y') {
//     Please input the Actor part
        get_actor(name, actor);
```

```
        Actor p_actor(name[0], name[1], name[2], name[3], name[4], name[5],
                name[6], name[7], name[8], name[9], name[10],
                name[11], name[12], name[13], name[14], name[15]);
        p_actor.persist(actor);
        p_actor.show();
    }    else
       p_people.persist(people);
       num++;
       /***  Actor key is Stagename        ****/
    printf("\n  Do you want to set up more objects?(Y/N)  \n");
    answer[0]='\0';
    getchar();
    };

    printf("      \nDisplay ending   \n\n");
    p_oodb1 -> commitT();
    printf("      New DB has been commited Successfully!\n\n");
}
```

```
*****************************
* Result 15                 *
*****************************

Commit failed, bus error.
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      pinfo.cc                                           */
/*    Date      :     05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-016                                             */
/*    Description :  Create a sub-class which is multiply inherited      */
/*                   from two super-classes.  These two super-classes    */
/*                   are, in turn derived from the same super-super-class */
/*    Requirements:  The attributes from both super-classes should be    */
/*              inherited to the sub-class.                              */
/*    Expected Results:   The dictionary should content all the class    */
/*                    relationships and attribute type information       */
/*                                                                       */
/*      C++ API commands used:                                           */
/*             People *p_ptr    = p_actor;                               */
/*           TYPEINFO &tyinfo = p_actor->oodb_typeof();                  */
/*           TYPEINFO &type_b = p_ptr->oodb_typeof();                    */
/*           tyinfo.describeall();                                       */
/*                                                                       */
/*      Testing Results : The dictionary entry for ADIRECTOR contains all */
/*                        the attributes inherited from ACTOR and DIRECTOR */
/*                        but attributes inherited from super-super-class */
/*                        PEOPLE are duplicated.                         */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>
#include     "schema.h"

OODB *p_oodb = new OODB("speckle:8000");

retrieveinfo()
{
    int           i;
    int       number;
    char     answer;
    char*    key = new char[30];

    number = 0;
    People *p_people = new People[1];
    Actor  *p_actor  = new Actor[1];
    ADirector *p_adirector = new ADirector[1];
    p_oodb -> beginT();

    TYPEINFO &tyinfo = p_adirector->oodb_typeof();
    printf( " ***** tyinfo.describeall();\n ");
    tyinfo.describeall();

    printf("\n\n Successfully!\n\n");
    p_oodb -> abortT();
    return(0);
}
```

```
*****************************
*  Result 16                *
*****************************

Following INFO from  "TYPEINFO &tyinfo = p_adctor ->oodb_typeof();"

{Type ADirector: 124 bytes, defined in file ./schema.h at line 156
 Derived from:  public Actor, public Director
  {Type Actor: 88 bytes, defined in file ./schema.h at line 78
   Derived from: virtual public People
    {Type Poeple:  56 bytes, defined in file ./schema.h at line 57
      Members of class people:
      id_name: [q=V2 type=t8s_String]
      pcode: [q=V2 type=t8s_String]
      lastnm: [q=V2 type=t8s_String]
      firstnm: [q=V2 type=t8s_String]
      . . .
       }

        Members of class Actor:
        stagename: [q=V2 type=t8s_String]
        roles: [q=V2 type8s_String]
        . . .
        }
    {Type Director: 84 bytes, defined in file ./schema.h at line 137
     Derived from: virtual public People
      {Type People:  56 bytes, defined in file ./schema.h at line 57
      Members of class people:
      id_name: [q=V2 type=t8s_String]
      pcode: [q=V2 type=t8s_String]
      lastnm: [q=V2 type=t8s_String]
      firstnm: [q=V2 type=t8s_String]
      . . .
        }
         Members of class Director:
           Company: [q=V3 type=t8s_String]
           Movies: [q=V3 type=t8s_String]
            . . .
         }
   Members of class Adirector:
       stagecode: [q=V3 type=t8s_String]
        . . .
      }
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      version1.cc                                        */
/*    Date      :     05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version   :     1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-019                                             */
/*    Description : Create a new version from the class PEOPLE.          */
/*    Requirements: Objects exist to be updated to create a new version  */
/*    Expected Results:   Display the new version.                       */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              p_people->AddExtension("V.0.2");                         */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              persist("my_object")                                     */
/*                                                                       */
/*      Testing Results : OK. New version exist.                         */
/*                                                                       */
/*************************************************************************/


#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>
#include     "schema.h"

OODB *p_oodb3 = new OODB("speckle:8000");

versionAdd()
{
    s_String*  name;
    char*   people = new char[maxFieldWidth];
    char*   actor = new char[maxFieldWidth];
    char*   answer;
    int      num, i;
    People *p_people = new People;
    Actor  *p_actor  = new Actor;
    name  = new s_String[maxFieldNum];
    answer = new char[3];
    answer[0] = '\0';
    num = 0;
    p_oodb3 -> beginT();

    p_people->AddExtension("V.o.2");

//  create objects with Version "V.0.2"

    for ( i=0; i<=maxFieldNum; i++) name[i] = new char[maxFieldWidth];
    printf("Do you want to set up PEOPLE objects?(Y/N)  \n");
    while ((answer[0] = getchar()) == 'y')   {
    get_data(name , people);
    p_people->create(name);
//       p_people->persist(people);
       p_people->show();
    cout << " Is " << name[0] << " a Actor?(Y/N)";
```

```
    cin >> answer;
    if ( answer[0] == 'y') {
         get_actor(name, actor);
         p_actor->create(name);
         p_actor->persist(actor);
         p_actor->show();
    };
    p_people->persist(people);
       num++;
       /***  Actor key is Stagename        ****/
    printf("\n  Do you want to set up more objects?(Y/N)  \n");
    answer[0]='\0';
    getchar();
    };
    printf("      \nDisplay ending   \n\n");
    p_oodb3 -> commitT();
    printf("      New DB has been commited Successfully!\n\n");
}

*********************************
*  Result 19                    *
*********************************

-----------------
Version "V.0.2":

id_name :    Chris
pcode      :    875
dcode      :    DP-1
Dow_start_end  :    1912-1999
last_name    :    Chris
first_name   :   John
dob        :    1900
dod        :    1999
origin     :    American
notes      :    pp
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                   for                                 */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      version.cc                                         */
/*    Date     :      05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-020                                             */
/*    Description : Fetch newly versioned objects and show version.      */
/*    Requirements:  Create a new version (do Test Number BDBF-19) and   */
/*                   retrieve the newly versioned object and display     */
/*                   the new version number.                             */
/*    Expected Results:   Objects with new version indication.           */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*          p_people->ShowExtensions();                                  */
/*                                                                       */
/*      Testing Results : OK.  Object with new version indication.       */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>

#include     "schema.h"

OODB *p_oodb1 = new OODB("speckle:8000");

versionShow()
{
    s_String*  name;
    int           i;
    int       number;
    char     answer;
    char*    key = new char[30];

    printf("==Following is going to retrieve the data from people DB==\n\n\n");
    printf("==Name of the classes is People  \n");
    number = 0;
    name  = new s_String[12];
    People *p_people;   //   = new People;
//    Actor  *p_actor;    = new Actor;
    p_oodb1 -> beginT();

    for ( i=0; i<=11; i++) name[i] = new char[30];
    printf("Do you want to retrieve PEOPLE OBJECTS? (y/n) \n");
    cin >> answer;
    key[0] = '\0';

    while ( answer == 'y') {
    printf("Do you want to retrieve specific OBJECTS? (y/n) \n");
    cin >> answer;
       while(answer == 'y'){
         printf("Please, give the key: NAME of the PEOPLE\n");
           cin >> key    ;
```

```
          cout <<" oodb_dd_instance->describe();\n" << flush;
        if((p_people = (People*) p_oodb1->fetch(key)) != NULL) {
        p_people -> show();
        p_people->ShowExtensions();
        } else {
          printf("Sorry, %s is not existing in DB! \n", key); };
          printf("Do you want to retrieve  more OBJECTS? (y/n) \n");
        cin >> answer;
      };
    printf("Do you want to quit the session(y/n)? \n");
      cin >> answer;
    if (answer == 'y') answer = 'n';
    else { printf("\n\n------------------------\n\n");
        answer = 'y';
    };
    };
    printf("\n\n Successfully!\n\n");
    p_oodb1 -> abortT();
    return(0);
}

****************************************
*  Result 20                           *
****************************************

Please, give the key: NAME of the PEOPLE
  Chris
      Display:

id_name :    Chris
pcode      :    871
dcode      :    DP-1
Dow_start_end  :    1912-1999
last_name    :    Chris
first_name   :   John
dob        :    1900
dod        :    1999
origin     :    American
notes      :    pp
Version : V.0.2
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      version_old.cc                                     */
/*    Date      :     05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-021                                             */
/*    Description : Fetch the old versioned objects which have been      */
/*                  newly versioned.                                     */
/*    Requirements:  Create a new version (do Test Number BDBF-19) and   */
/*                   retrieve the old versioned object.                  */
/*    Expected Results:   Objects with old version indication.           */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              OODB *my_oodb(hostname:port)                             */
/*              beginTransaction()                                       */
/*          p_people->ShowExtensions();                                  */
/*                                                                       */
/*      Testing Results : NO.  Cannot fetch the old versioned objects.   */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>

#include     "schema.h"

OODB *p_oodb1 = new OODB("speckle:8000");

versionShow()
{
    s_String*  name;
    int           i;
    int       number;
    char     answer;
    char*    key = new char[30];

    printf("==Following is going to retrieve the data from people DB==\n\n\n");
    printf("==Name of the classes is People  \n");
    number = 0;
    name  = new s_String[12];
    People *p_people;   //   = new People;
//    Actor  *p_actor;    = new Actor;
    p_oodb1 -> beginT();

    for ( i=0; i<=11; i++) name[i] = new char[30];
    printf("Do you want to retrieve PEOPLE OBJECTS? (y/n) \n");
    cin >> answer;
    key[0] = '\0';

    while ( answer == 'y') {
    printf("Do you want to retrieve specific OBJECTS? (y/n) \n");
    cin >> answer;
       while(answer == 'y'){
         printf("Please, give the key: NAME of the PEOPLE\n");
           cin >> key    ;
```

```
          cout <<" oodb_dd_instance->describe();\n" << flush;
        if((p_people = (People*) p_oodb1->fetch(key)) != NULL) {
        p_people -> show();
        p_people->ShowExtensions();
        } else {
          printf("Sorry, %s is not existing in DB! \n", key); };
          printf("Do you want to retrieve  more OBJECTS? (y/n) \n");
        cin >> answer;
      };
    printf("Do you want to quit the session(y/n)? \n");
      cin >> answer;
    if (answer == 'y') answer = 'n';
    else { printf("\n\n------------------------\n\n");
        answer = 'y';
    };
    };
    printf("\n\n Successfully!\n\n");
    p_oodb1 -> abortT();
    return(0);
}
```

```
**************************************
*  Result 21                         *
**************************************
```

The old version cannot be fetched.

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                   for                                 */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      pfetch_con1.cc                                     */
/*    Date      :     05/31/94                                           */
/*    Modifing  :     06/26/94                                           */
/*    Version   :     1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-022                                             */
/*    Description : Test for concurrency control when both clients       */
/*                  are trying to read the same object.                  */
/*    Requirements:  Start running T1: a "fetch object" program first, do */
/*                   not commit but, at another window initiate T2: another*/
/*                   "fetch object" program.  Commit T2 then T1.         */
/*    Expected Results:   No standard way for concurrency control.       */
/*                        Expect both programs can read the data, but one */
/*                        of the transaction will be locked at commit time.*/
/*                                                                       */
/*      C++ API commands used:                                           */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              fetch("sequence_No")                                     */
/*              abortT()                                                 */
/*                                                                       */
/*      Testing Results : Both T1 and T2 able to read object, but T1     */
/*                        cannot be committed.                           */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>

#include     "schema.h"

browse()
{
    s_String*  name;
    int           i;
    int      num, number;
    int      start_num;
    int      total;
    char     answer;
    int       blank;
    char*    key = new char[30];
    char*    movie = new char[40];

    printf("==Following is going to retrieve the data from movie DB==\n\n\n");
    printf("==Name of the classes is People\n");
    number = 0;
    OODB *p_oodb1 = new OODB("speckle:8000");
    name  = new s_String[12];
    People  *p_people;
    for ( i=0; i<=11; i++) name[i] = new char[30];
    p_oodb1 -> beginT();
          movie[0] = '\0';
          number = 1;
          strcpy(movie,  genOid(1000 + number));
          while ((p_people = (People*) p_oodb1->fetch(movie)) != NULL) {
```

```
            number++;
            strcpy(movie,  genOid(1000 + number));
          };
          total = number-1;
          num = number;
          start_num = 1;
    printf("Do you want to retrieve OBJECTS? (y/n) \n");
    cin >> answer ;
    key[0] = '\0';
    while ( answer == 'y') {
          printf("\nThere are \"%d\" objects \n", total  );
          printf("\nDo you want them all?(y/n)    \n");
          cin >> answer;
          if ( answer != 'y') {
          printf("==How many objects do you want to retrieve? \n");
          cin >> num ;
          cout << "==Starting Object No.:   ";
          cin >> start_num;
          cout << "\n" << flush;
          num = num +  start_num;
          };
          for ( number = start_num; number < num ; number++){
                movie[0] = '\0';
                strcpy(movie,  genOid(number + 1000));
          if ((p_people = (People*) p_oodb1->fetch(movie)) != NULL) {
          printf("\n Object NO.  %d \n", number);
          p_people -> show();
          };

          printf("Hit any key to continue");
          blank = getchar( );
          };     /*  end of FOR  */
         cout<<  "Do you want to quit the session(y/n)? \n" ;
         cin >> answer;
         if (answer == 'y') answer = 'n';
         else { printf("\n\n\n\n");
              answer = 'y';
         };
    };
          p_oodb1 -> abortT();
}
```

```
*****************************************
*  Result 022                           *
*****************************************
```

The observed behaviors are as follows:
Start T1 with read and start T2 with reading the same object.
If T1 commits, T2 will be lockout.
If T2 commits, T1 will be lockout.
The lockout occurred at commit time. Since both T1 and T2 are reading, no modificiation to the data occured. The second commit got lockout, but the read operation is fulfilled.

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                    for                                */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      pfetch_con2.cc                                     */
/*    Date      :     05/31/94                                           */
/*    Modifing  :     06/26/94                                           */
/*    Version   :     1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-023                                             */
/*    Description : Test for concurrency control when one client is      */
/*                  retrieving, while the other client is trying to      */
/*                  update the same object.                              */
/*    Requirements:  Start running T1: an "update object" program.       */
/*                   At another window initiate T2: a "fetch object"     */
/*                   program.  Commit T2 first then T1.                  */
/*    Expected Results:   No standard way for concurrency control.       */
/*                        Expect one transaction will be locked out.     */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              fetch("sequence_No")                                     */
/*              abortT()                                                 */
/*                                                                       */
/*      Testing Results : When T2 commits, the update of T1 is locked.   */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>

#include     "schema.h"

browse()
{
    s_String*  name;
    int           i;
    int      num, number;
    int      start_num;
    int      total;
    char     answer;
    int       blank;
    char*    key = new char[30];
    char*    movie = new char[40];

    printf("==Following is going to retrieve the data from movie DB==\n\n\n");
    printf("==Name of the classes is People\n");
    number = 0;
    OODB *p_oodb1 = new OODB("speckle:8000");
    name  = new s_String[12];
    People  *p_people;
    for ( i=0; i<=11; i++) name[i] = new char[30];
    p_oodb1 -> beginT();
          movie[0] = '\0';
          number = 1;
          strcpy(movie,  genOid(1000 + number));
          while ((p_people = (People*) p_oodb1->fetch(movie)) != NULL) {
```

```
            number++;
            strcpy(movie,  genOid(1000 + number));
          };
          total = number-1;
          num = number;
          start_num = 1;
    printf("Do you want to retrieve OBJECTS? (y/n) \n");
    cin >> answer ;
    key[0] = '\0';
    while ( answer == 'y') {
          printf("\nThere are \"%d\" objects \n", total  );
          printf("\nDo you want them all?(y/n)    \n");
          cin >> answer;
          if ( answer != 'y') {
          printf("==How many objects do you want to retrieve? \n");
          cin >> num ;
          cout << "==Starting Object No.:   ";
          cin >> start_num;
          cout << "\n" << flush;
          num = num +  start_num;
          };
          for ( number = start_num; number < num ; number++){
                movie[0] = '\0';
                strcpy(movie,  genOid(number + 1000));
          if ((p_people = (People*) p_oodb1->fetch(movie)) != NULL) {
          printf("\n Object NO.  %d \n", number);
          p_people -> show();
          };

          printf("Hit any key to continue");
          blank = getchar( );
          };     /*  end of FOR  */
         cout<<  "Do you want to quit the session(y/n)? \n" ;
         cin >> answer;
         if (answer == 'y') answer = 'n';
         else { printf("\n\n\n\n");
              answer = 'y';
         };
    };
          p_oodb1 -> abortT();
}
```

```
**********************************
*  Result 023                    *
**********************************

The observed behaviors are as follows:
      T1 ->start to update, then
      T2 ->try to retrieve.
      T2 got the data and commit.
      T1 -> Updating has been done and try to commit, but failed.
      T2 has been committed.

      The error message with T1 is as following:

      OpenOODB file:kernel/exodus.c line:330 errno:786441
                  error id:esmLOCKCAUSEDDEADLOCK - lock reqeust causes deadlock
      OpenOODB file:kernel/asm.c line:592 errno:786441
                  error id:esmLOCKCAUSEDDEADLOCK - lock reqeust causes deadlock
      ASM_Client::Write_Object(): sm_SetObjectHeader Failed - 1.
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                   for                                 */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      update_con3.cc                                     */
/*    Date     :      05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-024                                             */
/*    Description :  Test for concurrency control when one client is     */
/*                   updating, while the other client is trying to       */
/*                   retrieve the same object.                           */
/*    Requirements:  Start running  T1: a "fetch object" program.        */
/*                   At another window initiate T2: an "update object"   */
/*                   program. Comit T2 first then T1.                    */
/*    Expected Result:    No standard way for concurrency control.       */
/*                        Expect the fetched object should not get       */
/*                        old (earlier versioned) data.                  */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              fetch("sequence_No")                                     */
/*              persist("my_object")                                     */
/*              commitT()                                                */
/*              abortT()                                                 */
/*                                                                       */
/*      Testing Results : If updated is commited first. the previous read */
/*                        will be the old data.                          */
/*                                                                       */
/*************************************************************************/

#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>

#include     "schema.h"

OODB *p_oodb1 = new OODB("speckle:8000");

void People::update_value(char *key)
{
    char value[maxFieldWidth];
    printf("======================\n\n");
    printf("Editing the Value      \n");
    printf("Original Record Key is \n\n");
    printf("id_name :    %s\n\n", id_name);
    strcpy(key, id_name);
    printf("Updating the record ----------\n\n\n");
    printf("Note : if no change, the original Value has to be retyped!!\n\n");
    printf("Field ( 1): pcode      :    %s\n", pcode     );
    printf("Input New Value        :    \n" );
    scanf(" %s", pcode  );
//    cin.getline(value, 81);
//    if (strcmp(value, "") != 0) strcpy(pcode, value);
    printf("\nField ( 2): dcode      :    %s\n", dcode     );
    printf("Input New Value        :    \n" );
    scanf(" %s", dcode  );
    printf("\nField ( 3): Dow_start_end  :    %s\n", dow_start_end );
```

```
    printf("Input New Value        :    \n" );
    scanf(" %s", dow_start_end  );

    printf("\nField ( 4): last_name      :    %s\n", last_name   );
    printf("Input New Value        :    \n" );
    scanf(" %s", last_name  );

    printf("\nField ( 5): first_name      :    %s\n", first_name   );
    printf("Input New Value        :    \n" );
    scanf(" %s", first_name  );

    printf("\nField ( 6): dob      :    %s\n", dob   );
    printf("Input New Value        :    \n" );
    scanf(" %s", dob  );

    printf("\nField ( 7): dod      :    %s\n", dod   );
    printf("Input New Value        :    \n" );
    scanf(" %s", dod  );

    printf("\nField ( 8): origin   :    %s\n", origin   );
    printf("Input New Value        :    \n" );
    scanf(" %s", origin );

    printf("\nField ( 9): notes    :    %s\n", notes    );
    printf("Input New Value        :    \n" );
    scanf(" %s", notes );

}

main()
{
    int       number;
    char     answer;
    char*    key = new char[maxFieldWidth];
    char*    people  = new char[maxFieldWidth];
    People *p_people = new People;

    printf("==Following is going to retrieve the data from people DB==\n\n\n");
    printf("==Name of the classes is People  \n");
    number = 0;
    p_oodb1 -> beginT();
    printf("Do you want to retrieve and update  OBJECTS? (y/n) \n");
    cin >> answer;
    key[0] = '\0';

 //   while ( answer == 'y') {
//    printf("Do you want to retrieve specific OBJECTS? (y/n) \n");
//    cin >> answer;
       while(answer == 'y'){
         printf("Please, give the key: NAME of the PEOPLE\n");
           cin >> key    ;
         if ((p_people = (People*) p_oodb1->fetch(key)) != NULL){
         p_people -> show();
         p_people -> update_value(people);
         p_people -> show();
         } else ("The NAME: %s record is not exisiting in the DB.\n", key);
           printf("Are you sure all of the values are right?(y/n) \n");
         cin >> answer;
         if ( answer == 'y') p_people->persist(people); else printf(" You can
modify it later on !!\n");
           printf("Do you want to retrieve  more OBJECTS? (y/n) \n");
         cin >> answer;
      };
/*    printf("Do you want to quit the session(y/n)? \n");
      cin >> answer;
```

```
    if (answer == 'y') answer = 'n';
    else { printf("\n\n-------------------------\n\n");
         answer = 'y';
    };
    };
*/

    printf("\n\n Successfully!\n\n");
//    p_oodb1 -> abortT();
   p_oodb1 -> commitT();
    return(0);
}

*********************************
*  Result 024                   *
*********************************

The observed behaviors are as follows:
      T1 -> start to retrieve the object, then wait.
      T2 -> start to update the same object and try to commit.

      T2 will commit first and T1 will be lockout from committing.
      T1 have read the "old" data.
```

```
/*************************************************************************/
/*                              Testing Program                          */
/*                                   for                                 */
/*                              Texas Instruments OOODB                  */
/*                                                                       */
/*    File Name:      update_con.cc                                      */
/*    Date      :     05/31/94                                           */
/*    Modifing :      06/26/94                                           */
/*    Version  :      1.2                                                */
/*                                                                       */
/*    Test ID No. : BDBF-025                                             */
/*    Description :  Test for concurrency control when two clients       */
/*                   are trying to update the same object.               */
/*    Requirements:  Two clients T1 and T2 both are initiating an update */
/*                   program on same object.  T2 commit first.           */
/*    Expected Results:   No standard way for concurrency control.       */
/*                        Expect one transaction is updating and the other*/
/*                        should wait until lock is released.            */
/*                                                                       */
/*      C++ API commands used:                                           */
/*              beginTransaction()                                       */
/*              commitTransaction()                                      */
/*              fetch("sequence_No")                                     */
/*              persist("sequence_No")                                   */
/*              abortT()                                                 */
/*              commitT()                                                */
/*                                                                       */
/*      Testing Results : One of the transaction is lockout.             */
/*                                                                       */
/*************************************************************************/
#include     "OpenOODB.h"
#include     "Typeinfo.h"
#include     <libc.h>
#include       <stdio.h>
#include      <stdlib.h>
#include     <strings.h>
#include     <iostream.h>

#include     "schema.h"

OODB *p_oodb1 = new OODB("speckle:8000");

void People::update_value(char *key)
{
    char value[maxFieldWidth];
    printf("======================\n\n");
    printf("Editing the Value      \n");
    printf("Original Record Key is \n\n");
    printf("id_name :    %s\n\n", id_name);
    strcpy(key, id_name);
    printf("Updating the record ----------\n\n\n");
    printf("Note : if no change, the original Value has to be retyped!!\n\n");
    printf("Field ( 1): pcode      :    %s\n", pcode     );
    printf("Input New Value        :    \n" );
    scanf(" %s", pcode  );
//    cin.getline(value, 81);
//    if (strcmp(value, "") != 0) strcpy(pcode, value);
    printf("\nField ( 2): dcode      :    %s\n", dcode     );
    printf("Input New Value        :    \n" );
    scanf(" %s", dcode  );
    printf("\nField ( 3): Dow_start_end  :    %s\n", dow_start_end );
    printf("Input New Value        :    \n" );
    scanf(" %s", dow_start_end  );

    printf("\nField ( 4): last_name      :    %s\n", last_name   );
```

```
    printf("Input New Value        :    \n" );
    scanf(" %s", last_name  );

    printf("\nField ( 5): first_name      :    %s\n", first_name   );
    printf("Input New Value        :    \n" );
    scanf(" %s", first_name  );

    printf("\nField ( 6): dob       :    %s\n", dob   );
    printf("Input New Value        :    \n" );
    scanf(" %s", dob  );

    printf("\nField ( 7): dod       :    %s\n", dod   );
    printf("Input New Value        :    \n" );
    scanf(" %s", dod  );

    printf("\nField ( 8): origin    :    %s\n", origin   );
    printf("Input New Value        :    \n" );
    scanf(" %s", origin );

    printf("\nField ( 9): notes     :    %s\n", notes    );
    printf("Input New Value        :    \n" );
    scanf(" %s", notes );

}

main()
{
    int       number;
    char     answer;
    char*    key = new char[maxFieldWidth];
    char*    people  = new char[maxFieldWidth];
    People *p_people = new People;

    printf("==Following is going to retrieve the data from people DB==\n\n\n");
    printf("==Name of the classes is People  \n");
    number = 0;
    p_oodb1 -> beginT();
    printf("Do you want to retrieve and update  OBJECTS? (y/n) \n");
    cin >> answer;
    key[0] = '\0';

 //   while ( answer == 'y') {
//    printf("Do you want to retrieve specific OBJECTS? (y/n) \n");
//    cin >> answer;
       while(answer == 'y'){
          printf("Please, give the key: NAME of the PEOPLE\n");
            cin >> key    ;
          if ((p_people = (People*) p_oodb1->fetch(key)) != NULL){
          p_people -> show();
          p_people -> update_value(people);
          p_people -> show();
          } else ("The NAME: %s record is not exisiting in the DB.\n", key);
            printf("Are you sure all of the values are right?(y/n) \n");
          cin >> answer;
          if ( answer == 'y') p_people->persist(people); else printf(" You can
modify it later on !!\n");
            printf("Do you want to retrieve  more OBJECTS? (y/n) \n");
          cin >> answer;
      };
/*    printf("Do you want to quit the session(y/n)? \n");
      cin >> answer;
    if (answer == 'y') answer = 'n';
    else { printf("\n\n-------------------------\n\n");
          answer = 'y';
     };
```

```
    };
*/

    printf("\n\n Successfully!\n\n");
//    p_oodb1 -> abortT();
   p_oodb1 -> commitT();
    return(0);
}

*********************************************
*  Result 025                               *
*********************************************

The observed behaviors are as follows:
Case 1:
      T1 ->start to update, then
      T2 -> start to update.
      T1 ->try to commit, but has to be waiting T2.

      T2 ->try to commit, T2's request causes deadlock, system failed, then
      T1 successfully committed.

      Following are the message provided by system:

      OpenOODB file:kernel/exodus.c line:330 errno:786441
             error id:esmLOCKCAUSEDDEADLOCK - lock reqeust causes deadlock
      OpenOODB file:kernel/asm.c line:592 errno:786441
             error id:esmLOCKCAUSEDDEADLOCK - lock reqeust causes deadlock
      ASM_Client::Write_Object(): sm_SetObjectHeader Failed - 1.

Case 2:
       T1 ->start to update, then
       T2 -> start to update.
       T2 ->try to commit, but has to be waiting T1.

       T1 ->try to commit, T1's request causes deadlock, system failed, then
       T2 successfully committed.
```